455

Linux Essentials

www.4linux.com.br

Conteúdo

GNU/Linux: A Origem

2.1 Conhecendo um Novo Mundo

Utilizar um sistema GNU/Linux é muito mais do que optar por uma solução isenta de custos de licença. É usufruir de uma filosofia que antecedeu o software proprietário, e que permitiu, por exemplo, que a Internet crescesse de forma livre como a conhecemos hoje. Como usuário de Software Livre, precisamos compreender um pouco mais sobre essa ideologia e como ela promoveu o surgimento das várias distribuições.

O sistema GNU/Linux é frequentemente chamado apenas pelo seu segundo nome: Linux. Entretanto, essa designação não faz justiça a todos os desenvolvedores que vêm construindo o sistema operacional como um todo.

GNU, que é um acrônimo recursivo de GNU's Not Unix. Trata-se, de um grupo que foi fundado em 1984 por seu idealizador, Richard Stallman, com o intuito de criar um sistema operacional "Unix-like" desprovido de amarras e travas ao seu uso. Os desenvolvedores GNU criaram uma série de programas básicos para um sistema operacional funcional, como editores de texto e compiladores. Entretanto, havia um pedaço de código essencial, que ainda não tinha sido criado: o kernel.

Em 1991, um jovem finlandês chamado **Linus Torvalds** disponibilizou para o mundo a primeira versão do Linux, um kernel "Unix-like". A partir desse ponto, foi possível unir o kernel Linux com os softwares GNU, originando o sistema operacional que chamamos de GNU/Linux.

O mundo GNU/Linux não é apenas um conjunto de programas. Ele traz consigo

uma filosofia de Mundo Livre e colaborativo, no qual as pessoas podem utilizar esses softwares irrestritamente, acima de tudo, aprender com eles, uma vez que seu código fonte deve ser disponível a todos que queiram melhorá-lo ou apenas aprender com ele. Para que esse mundo continue livre, **Richard Stallman fundou a "FSF - Free Software Foundation"**, que criou e mantém a licença **"GNU GPL - GNU General Public License"**. Esta licença define, de modo simplificado, que o "Software" deve respeitar quatro princípios básicos, aqui chamados de liberdades. São elas:

- **Liberdade 0** - liberdade para rodar o programa para quaisquer propósitos;
- **Liberdade 1** - liberdade para estudar como o programa trabalha e adaptá-lo às suas necessidades. Ter acesso ao código fonte é essencial para isso;
- **Liberdade 2** - liberdade de redistribuir cópias de forma que você possa ajudar outras pessoas;
- **Liberdade 3** - liberdade para melhorar o programa e disponibilizar as melhorias para o público, de forma que toda a comunidade possa se beneficiar. Ter acesso ao código fonte é essencial também para isso.

Atualmente a GPL está disponível em três versões, GPLv1, GPLv2 e GPLv3. Fique por dentro de suas diferenças em:

 http://www.gnu.org/licenses/gpl.html

Como usar as licenças GPL:

 http://www.gnu.org/licenses/gpl-howto.pt-br.html

Para mais informações a respeito do kernel - Linux - podem ser obtidas no site oficial de seus mantenedores:

http://www.kernel.org

Informações sobre os projetos GNU e FSF podem ser obtidas nos seus respectivos sites:

 http://www.gnu.org http://www.fsf.org

2.1.1 Distribuições GNU/Linux

Você já deve ter ouvido falar em Debian, RedHat, Slackware, Suse, Mandriva, Ubuntu, CentOS dentre outras. Mas, o que realmente é isso? O que são todos esses nomes? Essas são distribuições GNU/Linux. Uma distribuição nada mais é do que o kernel Linux, softwares GNU e outros outros aplicativos que são desenvolvidos por outras comunidades ou grupos, reunidos em um sistema operacional que tem peculiaridades que o diferencia de outros sistemas operacionais GNU/Linux, fazendo-os únicos.

Mas, por que existem tantas distribuições? Justamente porque se você não se identifica com nenhuma delas, você é livre para fazer a sua própria. Por exemplo, em 1993, um rapaz chamado Patrick Volkerding, juntou o kernel e vários outros aplicativos em uma distribuição chamada Slackware, que foi a primeira a ser distribuída em CD. A partir desse ponto, foram surgindo diversas outras distribuições que de alguma forma diferiam da filosofia do Slackware: como Debian ou RedHat.

Atualmente existem centenas de distribuições, algumas mais famosas que outras. Em sua maioria, as distribuições GNU/Linux são mantidas por grandes comunidades de colaboradores, entretanto, há outras que são mantidas por empresas. Dessa forma, podemos dividir as **"distros"**, abreviação bastante utilizada na comunidade e que se refere às distribuições, em duas categorias básicas:

Distribuições Livres de Custos

Mantidas por comunidades de colaboradores sem fins lucrativos.

Exemplos são: Debian, Slackware, Gentoo, Knoppix e CentOS, entre outras.

Distribuições Corporativas

Mantidas por empresas que vendem o suporte ao seu sistema. Exemplos: RedHat, Ubuntu, Suse e Mandriva.

É a liberdade do software, garantida pela licença GPL, que perpetua o respeito dos direitos definidos pela FSF. Isso porque, pela definição de Software Livre, nunca, em hipótese alguma, é permitido que o código fonte seja negado ao cliente, ao receptor do Software. Assim, por mais que uma empresa queira cobrar por suas versões

do sistema GNU/Linux, enquanto ela estiver utilizando softwares licenciados sob a licença GPL, ela será obrigada a distribuir o código fonte dos programas.

Dentro do conjunto de Distribuições, podemos dividí-las novamente em duas outras categorias:

Distribuições Convencionais

São distribuídas da forma tradicional, ou seja, uma ou mais mídias que são utilizadas para instalar o sistema no disco rígido;

Distribuições Live

São distribuídas em mídias com o intuito de rodarem a partir delas, sem a necessidade de serem instaladas no HD. As distribuições "Live" ficaram famosas pois têm a intenção de fornecer um sistema GNU/Linux totalmente funcional, de forma fácil e sem colocar em risco o sistema operacional original da máquina. O fator que favoreceu essa abordagem é que, em uma distribuição "Live" praticamente todos os componentes já vêm configurados, funcionando e com interfaces agradáveis aos usuários finais. Exemplos desse tipo de distribuição são o "Knoppix", "Ubuntu"entre outras.

Para entender um pouco mais sobre distribuições, é necessário lembrar de mais duas características:

Distribuições From Scratch

São desenvolvidas do zero, ou seja, utilizam um kernel Linux, alguns programas GNU e a grande maioria das suas particularidades é desenvolvida especificamente para ela. Exemplos: Debian, RedHat, Gentoo, Slackware

Distribuições Provenientes (Baseadas)

Aproveitam ferramentas e bases já desenvolvidas por outras distribuições. Distribuições baseadas usam distribuições "From Scratch"para alcançar seus objetivos mais rápido, dando maior atenção ao propósito da distribuição. Exemplos: Ubuntu, DreamLinux, Kubuntu, Slax e Linux Mint

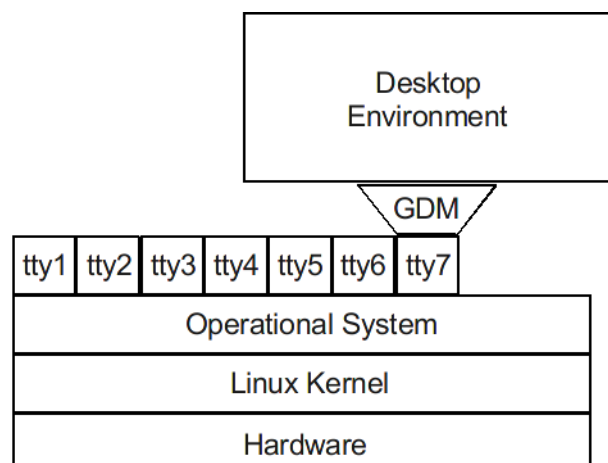
455

Linux Essentials

www.4linux.com.br

Primeiros Passos

3.1 Introdução teórica

A figura abaixo procura demonstrar como o Sistema Operacional GNU/Linux se organiza em “layers” - camadas:

É importante entender cada uma dessas camadas para compreender o conjunto que chamamos de **“Sistema Operacional”.**

Vamos descrever cada uma delas:

Hardware - Dispositivos que estão disponíveis para o uso do sistema, tais como cd-rom, placa de rede, controladora “SCSI” entre outros;

Kernel - O núcleo do sistema operacional, essa “layer” é quem faz todas as interações com o **hardware** da máquina, interpretando requisições feitas pelas camadas acima desta;

Sistema Operacional - Essa “layer” tem como função auxiliar e abrigar todos os aplicativos das camadas superiores. Segundo **Linus Torvalds** essa “layer” não deve ser notada pelo usuário final;

ttyN - Terminais Virtuais onde são executados os comandos e definidas as configurações. As **”tty’s”** interpretam os comandos dados por um humano e convertem os mesmos para uma linguagem que a máquina entenda;

DM - A “layer” de **“Display Manager”** é responsável por gerenciar os “logins” - validação de usuários - na interface gráfica e escolher o tipo de ambiente gráfico que deve ser executado;

Desktop Environment - Mais conhecido como **Ambiente de Trabalho,** é responsável por abrigar todos os programas que necessitam de um ambiente gráfico para funcionar.

3.1.1 Terminal Virtual

O GNU/Linux faz uso de sua característica multi-usuário, ou seja, suporta vários usuários conectados ao mesmo tempo, usando os “terminais virtuais”.

Um terminal virtual é uma segunda seção de trabalho completamente independente de outras e que pode ser acessado no computador local ou remotamente, utilizando os programas “telnet”, “rsh”, “rlogin”, “rdesktop”, “vnc”, “ssh”, etc. Nos dias de hoje, o acesso remoto é muito importante. A qualquer distância que se esteja do cliente, é possível atendê-lo.

No GNU/Linux é possível, em modo texto, acessar outros terminais virtuais, segurando a tecla “ALT” e pressionando uma das teclas de F1 até F6. Cada tecla tem

função correspondente a um número de terminal do 1 ao 6. Esse é o comportamento padrão e, pode ser mudado. (o sétimo, por “default”, é usado pelo ambiente gráfico - “Xorg”)

O GNU/Linux possui mais de 63 terminais virtuais, mas deles, apenas 6 estão disponíveis, inicialmente por motivos de economia de memória “RAM”. Se você estiver usando o modo gráfico, deve segurar “Ctrl + Alt” enquanto pressiona uma tecla de atalho de F1 a F6.

Um exemplo prático: se você estiver utilizando o sistema no terminal 1, pressione “Ctrl + Alt + F2”, e veja na primeira linha, nome e versão do sistema operacional, nome da máquina e o terminal no qual você está. Você pode utilizar quantos terminais quiser, do F1 ao F6 (inclusive utilizando o X) e pode ficar “saltando” de terminal para terminal.

3.1.2 Logon

Logon é a entrada do usuário, seja “root” ou comum, onde deve ser digitado seu nome de usuário e logo depois sua senha. Caso você digite algo de forma errada, irá aparecer uma mensagem de erro e você não será logado – autenticado - no sistema.

É importante perceber que quando se digita a senha, não aparece nenhum retorno, como os famosos asteriscos. O objetivo é evitar que um observador mais curioso seja capaz de contar quantos caracteres sua senha possui.

3.1.3 Shell

No Mundo GNU/Linux, utilizamos o **Shell,** que funciona como interpretador de comandos. Ele é a interface entre o usuário e o kernel do sistema e por meio dele,

podemos digitar os comandos. O ”Shell” padrão do GNU/Linux é o **Bash**. Entretanto existem também outras interfaces, como, por exemplo, “csh”, “tcsh”, “ksh” e “zsh”.

O kernel é a parte mais próxima do hardware do computador. É o núcleo do Sistema Operacional. Se seu **GNU/Linux** estiver com problemas, não chute seu computador, a culpa não é dele.

O local onde o comando será digitado é marcado por um “traço piscante” na tela, chamado de “cursor”. Tanto em “Shell” texto como em “Shell” gráfico é necessário o uso do “cursor” para saber onde devemos iniciar a digitação de textos e nos orientarmos quanto à posição na tela.

Popularmente conhecido como linha de comando, o “Shell” interpreta a ação do usuário através das instruções digitadas. Estas instruções poderão ser executadas por dois níveis de usuários, com permissões diferentes. São eles:

Super usuário: Popularmente conhecido como **"root"**. Não se engane, **"root"** não é de raiz, da língua inglesa. O usuário "root"é o administrador do sistema, e seu diretório (pasta) padrão é o "/root", diferentemente dos demais usuários que ficam dentro do "/home". No próximo capítulo falaremos mais sobre a instrutura de diretórios do GNU/LINUX. o "Shell"de uma usuário "root"é diferencia do "Shell"de um usuário comun, pois antes do cursor ele é identificado com **"#"** (jogo-da-velha).

Usuário comum: É qualquer usuário do sistema que não seja “root” e não tenha poderes administrativos no sistema. Como já havíamos dito anteriormente, o diretório padrão para os usuários é o **“/home”.** Antes do cursor, o “Shell” de um usuário comum é identificado com “$” (cifrão).

Existem muitas funcionalidades no “Shell”, uma delas é retornar comandos que já foram digitados anteriormente. Para fazer isso é só pressionar as teclas “seta para cima” e “seta para baixo” para ter acesso ao histórico de comandos. Inclusive o nome do programa responsável por manter essa lista é **“history”.**

Outra funcionalidade muito utilizada, serve para rolar a nossa tela de modo que possamos ir para cima ou para baixo, parecido com o “scroll” Para rolarmos a tela para

cima, segura-se a tecla “Shift” e pressionamos o “Page Up”. Para rolarmos a tela para baixo, segura-se a tecla “Shift” e pressionamos o “Page Down”. Isto é útil para ver textos que rolaram rapidamente para cima e saíram do nosso campo de visão.

A execução de comandos com poderes administrativos, exige que o nível do usuário comum seja alterado. Uma das formas de fazer isso é utilizando o comando **“su - Super User”**. Veja sua descrição abaixo:

- **su -** Para usar o comando “su” é necessário ter o “password” do administrador. Uma vez que o nível tenha sido mudado será possível executar qualquer comado com poderes de “root”.

Após se logar com usuário aluno, utilize o comando “su”:

```
1 $ su
```

Será pedido a senha do usuário root. Após efetuar a autenticação do usuário, o prompt mudará de “$” para “#” avisando que você está logado como administrador do sistema.

Existem dois comandos, "whoami"e "who am i"que lhe permite saber quem você é em determinado momento. A sequência de comandos abaixo esclarece o uso e finalidade destes dois comandos claramente:

```
1 # whoami
2 # who am i
```

O comando whoami indica quem você é no momento “root”. Se você utilizou o comando "su"para tornar-se outro usuário o comando "who am i"informa quem você realmente é “aluno”, pois foi com ele que você se logou na máquina antes de trocar de usuário.

Ele também pode ser utilizado para trocar de usuário, ele não pedirá a senha se você for usuário root:

```
1 # su - aluno
```

Com a opção “-” além de trocar de usuário, também carregará as variáveis locais do usuário:

```
1 $ su -
```

3.1.4 Configurações de Teclado no Console

Altere o “layout” de teclado padrão do sistema para ficar permanente:

```
1 # dpkg-reconfigure keyboard-configuration
```

E em seguida reinicie o serviço:

```
1 # /etc/init.d/keyboard-setup restart
```

Vamos realizar o mesmo procedimento no CentOS:

É possível utilizar o comando “loadkeys” para alterar o “layout” de teclado durante a sessão mas, essa alteração será temporária. Para trocar definitivamente o padrão de “layout” do teclado da máquina, altere o arquivo em /etc/sysconfig/keyboard.

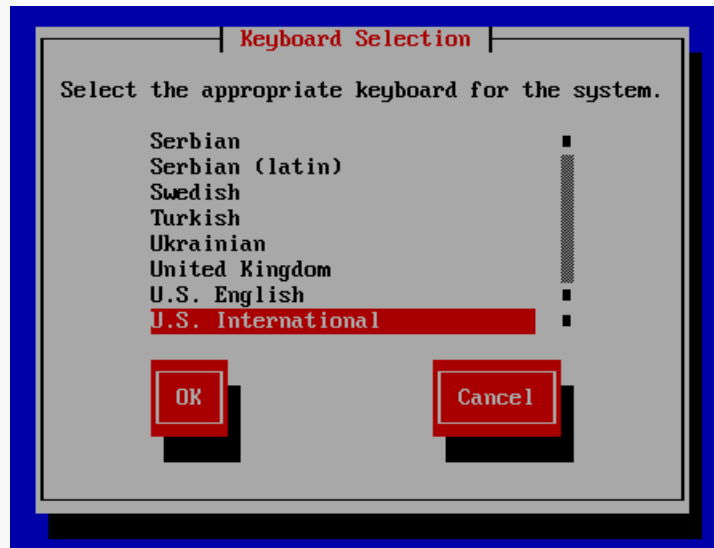
1) Altere o “layout” de teclado para utilizar o padrão brasileiro:

```
1 # loadkeys -d br-abnt2
```

2) Restaure o “layout” de teclado para o padrão “americano”:

```
1 # loadkeys -d us-acentos
```

Alterando o layout permanentemente:

```
1 # system-config-keyboard
```

Veja que ele altera o arquivo que armazena as configurações do teclado:

```
# cat /etc/sysconf/keyboard
KEYTABLE="us-acentos"
MODEL="pc105"
LAYOUT="us"
KEYBOARDTYPE="pc"
VARIANT="intl"
OPTIONS="qwerty"
```

3.1.5 Configurações do mouse no Console

Para se utilizar o mouse em modo texto, basta instalar o software gpm.

```
1 # apt-get install gpm
```

```
1 # yum install gpm
```

3.1.6 Histórico de comandos

Comando history

O terminal do GNU/Linux permite que você guarde 500 comandos por padrão no Debian e 1000 comandos no CentOS.

```
1 # history
```

Comando fc

FC significa "Find Command" ou "Fix Command" pois ele executa as duas tarefas, encontrar e corrigir comandos. Para listar os comandos já digitados, guardados no history, digite:

```
# fc -l
```

Por padrão mostra os últimos 16 comandos. Para visualizar uma lista de comandos do 2 ao 6 faça:

```
# fc -l 2 6
```

Para visualizar os últimos 20 comandos:

```
# fc -l -20
```

Para visualizar todos os comandos desde o último começando com h:

```
# fc -l h
```

3.1.7 Logout

Logout é a saída do sistema. Ela é feita por um dos comandos abaixo:

```
$ logout
$ exit
$ <CTRL>+D
```

Ou quando o sistema é reiniciado ou desligado.

3.1.8 Desligando o Computador

Para desligar o computador, pode-se utilizar um dos comandos abaixo, sempre que se esteja com o nível de usuário “root”:

```
1 # shutdown -h now
2 # halt
3 # poweroff
```

A palavra “halt” vem do comando em “assembly” chamado “HTL”, que quer dizer “parada de processamento”. Assim, o GNU/Linux finalizará os programas e gravará os dados remanescentes na memória no disco rígido. Quando for mostrada a mensagem “power down”, pressione o botão “POWER” em seu gabinete para desligar a alimentação de energia do computador. Nunca desligue o computador diretamente sem utilizar um dos comandos “shutdown”, “halt” ou “poweroff”, pois podem ocorrer perdas de dados ou falhas no sistema de arquivos de seu disco rígido, devido a programas abertos e dados ainda não gravados no disco. Os comandos “halt” e “poweroff” disparam uma série de procedimentos, como encerramento de serviços e desligamento de sistemas de arquivos, que são executados antes da máquina ser desligada.

Em computadores mais modernos o comando “halt” desliga completamente o computador, não sendo necessário pressionar o botão “Power”.

Salve seus trabalhos para não correr o risco de perdê-los durante o desligamento do computador. E se puder, tenha um “No-break”.

O comando **shutdown** tem a seguinte sintaxe:

```
1 # shutdown <ação> <tempo>
```

Onde: ação - o que você quer fazer, cujas opções são:

- h -> para desligar
- r -> para reiniciar.

tempo - tempo em minutos que você deseja para começar a executar a ação.

mensagem - Mensagem que você quer disparar para todos os terminais logados com o objetivo de avisar aos usuários que o sistema será desligado ou reiniciado.

Exemplos

Desligar agora:

```
1 # shutdown -h now
```

Ou

```
1 # shutdown -h 0
```

Desligar daqui 12 minutos notificando os demais usuários logagos com uma mensagem:

```
1 # shutdown -h 12  esta é minha mensagem de aviso
```

Para cancelar o shutdown:

```
1 # shutdown -c
```

3.1.9 Reiniciando o Computador

Reiniciar quer dizer ”Iniciar novamente o sistema”. Não é recomendável desligar e ligar constantemente o Computador pelo botão “ON/OFF” ou “RESET”. Por isso, existem recursos para reiniciar o sistema sem desligar o computador. No GNU/Linux você pode usar o comando **“reboot”**, **“shutdown -r now”** ou pressionar simultaneamente as teclas “Ctrl + Alt + Del” para reiniciar o sistema de forma segura.

Utilize comandos e não o botão liga/desliga. Prefira um dos métodos de reinicialização explicados acima e use o botão “reset” somente em último caso.

Exemplos

Reiniciar agora:

```
1 # shutdown -r now
```

Ou

```
1 # shutdown -r 0
```

Reiniciar daqui a 5 minutos com mensagem:

```
1 # shutdown -r 5   esta é minha mensagem de aviso
```

3.1.10 Sobrevivendo no Modo Texto

Vamos aprender agora alguns comandos essenciais para a nossa movimentação dentro do sistema.

O comando **"pwd"** exibe o diretório corrente. Ele é muito útil quando estamos navegando pelo sistema e não lembramos qual é o diretório atual.

```
1 # pwd
```

O comando **"ls"** é utilizado para listar o conteúdo dos diretórios. Se não for especificado nenhum diretório, ele irá mostrar o conteúdo daquele onde estamos no momento.

Listar o conteúdo do diretório atual:

```
1 # ls
```

O comando **"cd"** é utilizado para mudar o diretório atual de onde o usuário está.

Exemplos

Ir para o diretório "home" do usuário logado:

```
1 # cd
2 # cd ~
```

Ir para o início da árvore de diretórios, ou seja, o diretório "/":

```
1 # cd /
```

Ir para um diretório específico:

```
1 # cd /etc
```

Sobe um nível na árvore de diretórios:

```
1 # cd ..
```

Retorna ao diretório anterior:

```
1 # cd -
```

Entra em um diretório específico:

```
1 # cd /usr/share/doc
```

Sobe 2 níveis da árvore de diretórios:

```
1 # cd ../../
```

Atenção! Note a diferença entre caminhos absolutos e relativos:

Absolutos: /etc/ppp; /usr/share/doc; /lib/modules

Relativos: etc/ppp; ../doc; ../../usr;

Diretório . e ..

Fique esperto para conhecer as diferenças entre o "." e o ".." e o que eles representam para o sistema. Os comandos de movimentação muitas vezes são grandes alvos nas provas, uma boa interpretação desses comandos pode ser necessária, pois você pode precisar deles para resolver uma questão maior.

Atalhos do bash

A seguir, vamos testar algumas **funcionalidades da linha de comando.** Não é necessário se preocupar em decorá-los, com o passar do tempo, pegamos um pouco mais de prática:

- Pressione a tecla **"Back Space"** para apagar um caractere à esquerda do cursor;
- Pressione a tecla **"Delete"** para apagar o caractere à direita do cursor;
- Pressione a tecla **"Home"** para ir ao começo da linha de comando;
- Pressione a tecla **"End"** para ir ao final da linha de comando;
- Pressione as teclas **"Ctrl + A"** para mover o cursor para o início da linha de comandos;
- Pressione as teclas **"Ctrl + E"** para mover o cursor para o fim da linha de comandos;
- Pressione as teclas **"Ctrl + U"** para apagar o que estiver à esquerda do cursor. O conteúdo apagado é copiado e pode ser colado com "**Ctrl + y";**

- Pressione as teclas "**Ctrl + K**" para apagar o que estiver à direita do cursor. O conteúdo apagado é copiado e pode ser colado com **"Ctrl + y";**

- Pressione as teclas **"Ctrl + l"** para limpar a tela e manter a linha de comando na primeira linha. Mas se você der um **"Shift + Page Up"** você ainda consegue enxergar o conteúdo. O **"Ctrl + l"** é um atalho para o comando **"clear";**

- Pressione as teclas "**Ctrl + c**" para abrir uma nova linha de comando, na posição atual do cursor;

- Pressione as teclas "**Ctrl + d**" para sair do "**Shell**". Este é equivalente ao comando "exit";

- Pressione as teclas "**Ctrl + r**" para procurar "**x**" letra relacionada ao último comando digitado que tinha "x" letra como conteúdo do comando.

- Executar o último comando pressione: **"!!"**

- Executar um comando específico do histórico de comandos: **"!<numero>",** ou seja, **"!12"**

Obtendo ajuda

4.2 Introdução teórica

O ritmo de geração de conhecimento e informação tem sido vertiginoso nos últimos cinquenta anos, especialmente na área tecnológica. Por isso é fundamental saber onde buscar informações para manter-se sempre atualizado. Neste capítulo, vamos aprender a consultar as documentações existentes e como buscar informações sobre o que precisamos.

O Sistema Operacional GNU/Linux possui uma vasta biblioteca de documentação. Antes de recorrermos a ajuda de outras pessoas, devemos lembrar que podemos ter a respostas que precisamos no próprio sistema, bem a nossa frente, ao teclar de um simples comando. Essa documentação em grande parte dos casos é de extrema qualidade.

O GNU/Linux cresceu porque a comunidade contribui para o sistema e sua documentação. Essa comunidade não tem medo ou receio de compartilhar informações e disponibiliza o que foi desenvolvido no próprio sistema. É muito importante reforçar que no Software Livre, as pessoas nunca ocultam seu **"know-how",** ou seja, você pode perguntar a vontade, desde que saiba o que e onde perguntar.

A documentação do GNU/Linux pode ser vista também como fonte de conhecimento, onde pode-se aprender muito sobre cada um dos serviços e comandos disponíveis.

Essa ajuda é provida por meio dos manuais, as famosas **"Man Pages".**

Toda essa documentação que possuímos no sistema GNU/Linux está disponível no site: http://www.tldp.org (The Linux Documentation Project), o site oficial de documentações sobre GNU/Linux.

Um diferencial deste site é ter a documentação em vários idiomas e formatos: pdf, html, txt e outros.

Abaixo vamos começar a nos familiarizar com a documentação existente e as formas nas quais ela é apresentada.

4.3 Formas de Documentação

Existem diversas formas de se documentar um projeto, dentre elas temos os "How-to's", os manuais e as documentações.

4.3.1 How-to's

Os "How-to's" são documentos que focam uma necessidade específica, como montar um "firewall", instalar uma "webcam", configurar placas de som, configurar um servidor web e muitos outros. Normalmente esses documentos são instalados junto com suas respectivas aplicações ou podem ter um pacote específico para a documentação daquela aplicação. Os "how-to's" também são conhecidos como **"cook-books"** - livro de receitas.

O diretório de "How-to's" do GNU/Linux é o **"/usr/share/doc"**. Se desejamos saber como configurar um "firewall", podemos consultar os arquivos do diretório:

```
1  # cd /usr/share/doc/iptables/
```

Na Internet existem diversos sites de "how-to's" para GNU/Linux. Dentre eles o mais conhecido no Brasil é o "Viva o Linux", conhecido também como VOL:

Muitas vezes o uso de "how-to's" ou "cook-book's", não agrega um bom conhecimento, pois trata-se somente de uma lista de afazeres para chegar a um objetivo. Quando o software é atualizado, todo aquele conhecimento fica dependente de um novo "how-to".

4.3.2 Manuais

Diferente dos "How-to's" os manuais não vão te mostrar um passo a passo ou mesmo te dar uma lista de afazeres. O principal objetivo do manual é te mostrar como as funcionalidades daquele software podem ser usadas. Com o manual o aprendizado para a utilização da ferramenta é facilitado, já que o mesmo possui alguns exemplos de usabilidade. Esses manuais podem ser encontrados através do comando **"man",** o qual veremos ainda nesse capítulo, um pouco mais adiante.

4.3.3 Documentação

A palavra documentação é muito intensa. Quando falamos em documentar uma ferramenta, estamos na realidade abrangendo uma série de outros itens importantes, dentre eles os "How-to's" e os manuais. Com a documentação de um projeto é possível entender absolutamente tudo sobre o mesmo, ou seja, essa documentação deve mostrar todas as partes relacionadas ao projeto.

Podemos, por exemplo, citar a documentação de um projeto de rede, onde deve constar não só documentos como "how-to's" e manuais, mas sim todas as especificações dos componentes, bem como cabos, "switch's" e "routers" dentre outros detalhes muito importantes.

Como esse tipo de documentação é muito especifica, devemos consultar o site de cada projeto individualmente.

Existem diversos comandos de ajuda no GNU/Linux, vamos abordar cada um deles logo abaixo:

4.3.4 Comando help

O comando "help" provê ajuda para comandos internos do interpretador de comandos, ou seja, o comando "help" fornece ajuda rápida. Ele é muito útil para saber quais opções podem ser usadas com os comandos internos do interpretador de comandos (shell).

Para visualizar uma ajuda rápida para todos os comandos internos do sistema, podemos fazer da seguinte forma:

```
1  # help
```

Caso desejemos visualizar a ajuda rápida para somente um comando interno, usamos esta outra sintaxe:

```
1  # help [comando]
```

 O comando "help" somente mostra a ajuda para comandos internos.

```
1 # help type
```

O comando type mostra se cada nome de comando é um comando do UNIX, um comando interno, um alias, uma palavra-chave do shell ou uma função de shell definida.

Verifique o tipo do comando **help** que conheceremos a seguir:

```
1 # help help
```

Para comandos externos, o "help" aparece como parâmetro. Por exemplo:

```
1 # [comando] --help
```

Desse modo, caso desejemos visualizar uma ajuda rápida sobre um comando externo, devemos fazer da seguinte forma:

```
1 # ls --help
```

O parâmetro "–help" pode ser utilizado em qualquer comando para ter uma consulta rápida dos parâmetros que ele pode nos oferecer. É importante entender que "–help" é na verdade um parâmetro individual de cada comando, logo se um comando não tiver esse parâmetro existem outros meios para se obter ajuda. Não se esqueça de estudar as diferenças entre comandos internos e externos.

4.3.5 Comando apropos

O comando “apropos” é utilizado quando não se sabe qual documentação acessar para um determinado assunto, mostrando as “man pages” que contém a palavra-chave que foi especificada.

A sintaxe utilizada para usar o “apropos” é a seguinte:

```
1  # apropos [palavra-chave]
```

Imagine que você precise editar um arquivo, mas não sabe qual editor utilizar. Execute o apropos para procurar algum comando ou manual de um comando para edição:

```
1  # apropos editor
```

Uma forma equivalente ao “apropos” é usar o comando “man” juntamente com a opção “-k”:

```
1  # man -k editor
```

4.3.6 Comando whatis

O comando **“whatis”** tem basicamente a mesma função do comando “apropos”, porém as buscas do comando “whatis” são mais específicas. O “apropos” busca as páginas de manuais e descrições de maneira mais genérica. Se digitarmos a palavra “passwd” ele nos trará tudo que tiver “passwd”, seja como nome ou parte do nome do manual ou na descrição. Já o “whatis” nos trará somente o manual com nome exato da palavra pesquisada. A sintaxe utilizada no comando “whatis” é a seguinte:

```
1 # whatis [comando]
```

Você sabe que tem um programa chamado “vim”, mas não sabe o que ele faz?

```
1 # whatis vim
```

Uma forma equivalente ao “whatis” é usar o comando “man” juntamente com a opção “-f”:

```
1 # man -f vim
```

Para localizar as “man pages”, o comando “apropos” e “whatis” utilizam o mesmo banco de dados construído com o comando **“catman”** ou **“makewhatis”** (executado pelo administrador do sistema, “root”). Para construir o banco de dados do comando “apropos” e whatis devemos executar o comando abaixo:

Debian:

```
1 # catman
```

CentOS:

```
1 # makewhatis -v
```

Os comandos “apropos” e “whatis” utilizam a mesma base de dados, é importante perceber isso. catman (Debian) e makewhatis (CentOS)

4.3.7 Comando man

O comando **“man”** é o responsável por trazer os manuais mais completos sobre determinado comando, arquivo de configuração, bibliotecas, entre outros nos quais estamos trabalhando.

Os manuais do sistema são divididos nos seguintes níveis:

- **man 1 ->** Programas executáveis e comandos do “Shell”;
- **man 2 ->** Chamadas de sistema (funções providas pelo Kernel);
- **man 3 ->** Chamadas de bibliotecas (funções como bibliotecas do sistema);
- **man 4 ->** Arquivos de dispositivo (Localizados normalmente no “/dev”);
- **man 5 ->** Arquivos de configuração e convenções;
- **man 6 ->** Jogos;
- **man 7 ->** Variados (incluindo pacotes de macros e convenções);
- **man 8 ->** Comandos de administração do sistema (normalmente usado somente pelo root);
- **man 9 ->** Rotinas de Kernel.

É comum o exame cobrar mais dos níveis 1, 5 e 8 dos manuais! Então lembre-se de estudar binários, arquivos de configuração e comandos administrativos.

Sintaxe do comando “man”:

```
1 # man [comando]
```

ou

```
1 # man [seção] [comando]
```

Essas informações sobre as seções do comando “man” podem ser encontradas em seu próprio manual, digitando o comando “man man”.

Se for necessário visualizar o manual do comando “passwd”, podemos fazer da seguinte forma:

```
1 # man passwd
```

Para navegar pelo manual, o comando “man” abre um arquivo que está compactado na pasta **“/usr/share/man/man1”** para o **“passwd”**. Outros níveis de manuais, dependem do comando ou arquivo.

O “passwd” é conhecido no sistema GNU/Linux como um comando que adiciona ou modifica a senha do usuário e, também, como o arquivo de usuários do sistema (/etc/passwd).

Veremos agora o manual do arquivo de usuários “passwd”:

```
1 # man 5 passwd
```

Podemos consultar quais manuais estão disponíveis dentro do próprio diretório do man:

```
1 # ls /usr/share/man/
```

Dentro desse diretório é possível ver todas as divisões dos manuais: os níveis, os idiomas e mais. Todos os níveis de manuais possuem sua determinada introdução que pode ser vista com o comando:

```
1 # man <nível> intro
```

Podemos ver os manuais em diversos idiomas diferentes, desde que o pacote para o idioma escolhido esteja instalado. Se nosso sistema estiver instalado em português, o comando “man” irá trazer todas os manuais disponíveis em português.

Já se nosso sistema estiver em inglês é preciso usar o parâmetro **“-L pt_BR”,** para que possamos ver os manuais em nosso idioma:

```
1 # man -L pt_BR comando
```

É importante nesse ponto ressaltar que a documentação em nosso idioma depende de pessoas que ajudam a fazer a tradução para o português, se você quiser ajudar, acredite, você será muito bem vindo. Veja como ajudar com o comando:

```
1 # man 7 undocumented
```

Podemos ver que para visualizar o manual do arquivo de usuário “passwd” precisamos informar em qual nível de manual ele se encontra, pois já existe um “passwd” no nível 1, que é o comando, então ele aparece primeiro quando digitamos “man passwd” sem indicar o nível. Esse manual do arquivo “passwd” está compactado na pasta “/**usr/share/man/man5”.**

4.3.8 Comando info

As **“info pages”** são como as páginas de manuais, porém são utilizadas com navegação entre as páginas. Elas são acessadas pelo comando **“info”.** Este é útil quando já sabemos o nome do comando e só queremos saber qual sua respectiva função.

A navegação nas “info pages” é feita através de nomes marcados com um “*” (hipertextos) que, ao pressionarmos **“Enter”,** nos leva até a seção correspondente, e “Backspace” volta à página anterior. Algo parecido com a navegação na Internet.

Podemos também navegar pelas páginas com as teclas **“n” (next/próximo)**; **“p” (previous/anterior); “u” (up/sobe um nível).** Para sair do comando “info”, basta pressionar a tecla “q”.

Se for necessário exibir a lista de todos os manuais de comandos/programas disponíveis, execute o comando abaixo sem nenhum argumento. Assim:

```
1  # info
```

Para exibir as informações somente de um determinado comando, usaremos a seguinte sintaxe:

```
1  # info [comando]
```

Visualizar informações do comando vim:

```
1 # info vim
```

Alternativas para consulta

Para obter uma melhor visualização, duas ferramentas de documentação foram desenvolvidas:

yelp -> Ferramenta gráfica para visualização de manuais de aplicativos gráficos do GNOME; (fornecido pelo pacote yelp)

xman -> "Front-end" para o comando "man", que facilita a consulta das "man pages"; (fornecido pelo pacote x11-apps)

4.3.9 Comando whereis

O comando **"whereis"** é utilizado para mostrar a localização do binário do comando, do arquivo de configuração (caso exista), e a localização das páginas de manuais do determinado comando ou arquivo.

Para visualizarmos a localização destes dados para um determinado comando ou arquivo, utilizamos a seguinte sintaxe:

```
1 # whereis <comando>
```

ou

```
1 # whereis <arquivo>
```

Mostrar a localização do binário do comando, do arquivo de configuração (caso exista), e a localização das páginas de manuais do comando “vim”:

```
# whereis vim
```

4.3.10 Comando which

O comando **“which”** é bem semelhante ao comando “whereis”, entretanto este só mostra a localização do binário do comando.

Para visualizar a localização do binário do comando, utilizamos a seguinte sintaxe:

```
 # which <comando>
```

Localização do binário do comando “vi”:

```
# which vi
```

455

Linux Essentials

www.4linux.com.br

Conteúdo

Aprendendo Comandos do GNU/Linux

4.1 Introdução teórica

Comandos são instruções passadas ao computador para executar uma determinada tarefa. No mundo *NIX (GNU/Linux,Unix), o conceito de comandos é diferente do padrão MS-DOS. Um comando é qualquer arquivo executável, que pode ser ou não criado pelo usuário.

Uma das tantas vantagens do GNU/Linux é a variedade de comandos que ele oferece, afinal, para quem conhece comandos, a administração do sistema acaba se tornando um processo mais rápido.

O “Shell” é o responsável pela interação entre o usuário e o sistema operacional, interpretando os comandos.

É no “Shell” que os comandos são executados.

4.1.1 O comando ls

O comando “ls” possui muitos parâmetros, veremos aqui as opções mais utilizadas. A primeira delas é o “-l” que lista os arquivos ou diretórios de uma forma bem detalhada (quem criou, data de criação, tamanho, dono e grupo ao qual cada um pertence):

```
1  # ls -l /
2 drwxr-xr-x4 root root 1024 2007-01-15 23:17 boot
```

Veja que a saída desse comando é bem detalhada. Falando sobre os campos, para o primeiro caractere temos algumas opções:

```
1 d => indica que se trata de um diretório
2 l => indica que se trata de um "link" (como se fosse um atalho -
    também vamos falar sobre ele depois)
3 - => hífen, indica que se trata de um arquivo regular
4 c => indica que o arquivo é um dispositivo de caractere (sem buffer)
5 b => indica que o arquivo é um dispositivo de bloco (com buffer)
6 u => "sinônimo para o tipo c" indica que o arquivo é um dispositivo
    de caractere (sem buffer)
7 s => indica que o arquivo é um socket
8 p => indica que o arquivo é um fifo, named pipe
```

FIFO - Sigla para First In, First Out, que em inglês significa primeiro a entrar, primeiro a sair. São amplamente utilizados para implementar filas de espera. Os elementos vão sendo colocados no final da fila e retirados por ordem de chegada. Pipes (|) são um exemplo de implementação de FIFO.

Buffer - É uma região de memória temporária, usada para escrita e leitura de dados. Normalmente, os buffers são utilizados quando existe uma diferença entre a taxa em que os dados são recebidos e a taxa em que eles podem ser processados.

Socket - É um meio de comunicação por software entre um computador e outro. É uma combinação de um endereço IP, um protocolo e um número de porta do protocolo.

O campo “rwxr-xr-x” lista as permissões, enquanto os campos “root” indicam quem é o usuário e grupo dono desse diretório que, no nosso caso, é o administrador do

sistema, o usuário “root”. O número antes do dono indica o número de “hard links”, um assunto abordado apenas em cursos mais avançados.

O campo “1024” indica o tamanho do arquivo, e o campo “2007-01-15 23:17” informa a data e hora em que o diretório foi criado. Finalmente, no último campo temos o nome do arquivo ou diretório listado, que, no nosso exemplo, é o “boot”.

Com relação aos diretórios, é importante ressaltar que o tamanho mostrado não corresponde ao espaço ocupado pelo diretório e seus arquivos e subdiretórios. Esse espaço é aquele ocupado pela entrada no sistema de arquivos que corresponde ao diretório.

A opção **“-a”** lista todos arquivos, inclusive os ocultos:

```
1
2  # ls -a /root
3 ..aptitude.bashrc.profile .rnd.ssh.vmware
4 .. .bash_history .kde .qt root_161206 .viminfo .Xauthority
```

Veja que, da saída do comando anterior, alguns arquivos são iniciados por “.” (ponto). Esses arquivos são ocultos. No Linux, arquivos e diretórios ocultos são iniciados por um “.” (ponto). Listar arquivos de forma recursiva, ou seja, listar também os subdiretórios que estão dentro do diretório ”/”:

```
1  # ls -R /
```

Como listar os arquivos que terminam com a palavra “.conf” dentro do diretório “/etc”?

```
1 # ls /etc/*.conf
```

Como buscar no diretório raiz ”/ “ todos os diretórios que terminem com a letra “n”?

```
1 # ls -ld /*n
```

4.1.2 Criar arquivo

Para criar um arquivo, podemos simplesmente abrir um editor de texto e salvá-lo. Mas existem outras formas. Uma das formas mais simples é usando o comando **“touch”:**

```
1 # cd /srv
2 # touch procedimentos.txt
3 # touch contas.pdf contas2.pdf contas3.pdf contas4.pdf
```

4.1.3 Curingas

O significado da palavra curinga no dicionário é o seguinte: carta de baralho, que em certos jogos, muda de valor e colocação na sequência. No sistema GNU/Linux é bem parecida a utilização desse recurso. Os curingas são utilizados para especificar um ou mais arquivos ou diretórios.

Eles podem substituir uma palavra completa ou somente uma letra, seja para listar, copiar, apagar, etc. São usados cinco tipos de curingas no GNU/Linux:

```
1 * - Utilizado para um nome completo ou restante de um arquivo/diretó
    rio;
2 ? - Esse curinga pode substituir uma ou mais letras em determinada
    posição;
3 ! - exclui da operação
```

```
[padrão] - É utilizado para referência a uma faixa de caracteres de um arquivo/diretório.
[a-z][0-9] - Usado para trabalhar com caracteres de a até z seguidos de um caractere de 0 até 9.
[a,z][1,0] - Usado para trabalhar com os caracteres a e z seguidos de um caractere 1 ou 0 naquela posição.
[a-z,1,0] - Faz referência do intervalo de caracteres de a até z ou 1 ou 0 naquela posição.
[^abc] - Faz referência a qualquer caracter exceto a, b e c.
{padrão} - Expande e gera strings para pesquisa de padrões de um arquivo/diretório.
X{ab,01} - Faz referência a sequência de caracteres Xab ou X01.
X{a-e,10} - Faz referência a sequência de caracteres Xa Xb Xc Xd Xe X10
```

DICA: - A barra invertida serve para escapar um caracter especial, ela é conhecida também como “backslash”.

A diferença do método de expansão dos demais, é que a existência do arquivo ou diretório é opcional para resultado final. Isto é útil para a criação de diretórios.

Os 5 tipos de curingas mais utilizados **(*, ?, [ ], , !)** podem ser usados juntos. Vejamos alguns exemplos:

Vamos criar 5 arquivos no diretório “/srv” utilizando o método de expansão.

```
# cd /srv
# touch curriculo{1,2,3}.txt curriculo{4,5}.new
```

Podemos listá-los assim:

```
1 # ls
2   curriculo1.txt  curriculo2.txt  curriculo3.txt  curriculo4.new
      curriculo5.new
```

Vamos listar todos os arquivos do diretório "/srv". Podemos usar o curinga "*" para visualizar todos os arquivos do diretório:

```
1 # ls *
2  curriculo1.txt  curriculo2.txt  curriculo3.txt  curriculo4.new
     curriculo5.new
```

Para listarmos todos os arquivos do diretório "/srv" que tenham "new" no nome:

```
1 # ls *new*
2 curriculo4.new curriculo5.new
```

Listar todos os arquivos que começam com qualquer nome e terminam com ".txt":

```
1  # ls *.txt
2 curriculo1.txt curriculo2.txt curriculo3.txt procedimentos.txt
```

Listar todos os arquivos que começam com o nome "curriculo", tenham qualquer caractere no lugar do curinga, e terminem com ".txt":

```
1  # ls curriculo?.txt
2 curriculo1.txt curriculo2.txt curriculo3.txt
```

Para listar todos os arquivos que começam com o nome "curriculo", tenham qualquer caractere entre o número "1-3" no lugar da 4ª letra e terminem com ".txt". Neste

caso, se obtém uma filtragem mais exata, pois o curinga especifica qualquer caractere naquela posição e ”[ ]” especifica um intervalo de números ou letras que será usado:

```
1  # ls curriculo[1-3].txt
2 curriculo1.txt curriculo2.txt curriculo3.txt
```

Para listar todos .txt exceto o curriculo2.txt:

```
1 # ls curriculo[!2].txt
2 curriculo1.txt curriculo3.txt
```

Para listar os arquivos “curriculo4.new” e “curriculo5.new” podemos usar os seguintes métodos:

```
1  # ls *.new
2  # ls *new*
3  # ls curriculo?.new
4  # ls curriculo[4,5].*
5  # ls curriculo[4,5].new
```

Existem muitas outras sintaxes possíveis para obter o mesmo resultado. A mais indicada será sempre aquela que atender à necessidade com o menor esforço possível. A criatividade nesse momento conta muito. No exemplo anterior, a última forma resulta na busca mais específica. O que pretendemos é mostrar como visualizar mais de um arquivo de uma só vez. O uso de curingas é muito útil e pode ser utilizado em todas as ações do sistema operacional referentes aos arquivos e diretórios: copiar , apagar, mover e renomear.

4.1.4 Criando diretórios

O comando **“mkdir”** é utilizado para criar um diretório no sistema. Um diretório é uma pasta onde você guarda seus arquivos. Exemplo:

Criar o diretório “Suporte”:

```
1 # cd /srv
2 # mkdir Suporte
```

Criar o diretório “Financeiro” e o subdiretório “Contas a Pagar”:

```
1  # mkdir -p Financeiro/Contas\ a\ Pagar
```

A opção **“-p”** permite a criação de diretórios de forma recursiva. Para que um subdiretório exista, o seu diretório diretamente superior tem que existir. Portanto a criação de uma estrutura como “Rh/Processos/Cv’s” exigiria a execução de três comandos **“mkdir”.**

Algo como:

```
1 # mkdir Rh
2 # mkdir Rh/Processos
3 # mkdir Rh/Processos/Cv\'s
```

A opção “-p” permite que toda essa estrutura seja criada em uma única linha. Assim:

```
1 # mkdir -p Rh/Processos/Cv\'s
```

4.1.5 Removendo arquivos/diretórios

O comando **"rm"** é utilizado para apagar arquivos, diretórios e subdiretórios estejam eles vazios ou não.

Exemplos:

Remover os arquivos com extensão "txt":

```
# cd /srv
# ls
# rm curriculo?.txt
# ls
```

Remover o arquivo "curriculo4.new" pedindo confirmação:

```
 # rm -i curriculo4.new
rm: remover arquivo comum vazio 'curriculo4.new'?
```

A opção **"-i"** força a confirmação para remover o arquivo "curriculo4.new".

Remover o diretório "Rh":

```
 # rm -r Rh
```

A opção **"-r " ou "-R"** indica recursividade, ou seja, a remoção deverá ser do diretório Rh e de todo o seu conteúdo.

Observação: Muita atenção ao usar o comando “rm”! Uma vez que os arquivos e diretórios removidos não podem mais ser recuperados!

O comando **“rmdir”** é utilizado para remover diretórios vazios.

Exemplos:

Remover o diretório “Suporte”:

```
1 # rmdir Suporte
```

4.1.6 Copiar arquivos/diretórios

O comando **“cp”** serve para fazer cópias de arquivos e diretórios. Perceba que para lidar com diretórios a opção “-r” ou “-R” tem que ser usada:

```
1 # cp arquivo-origem arquivo-destino
2
3 # cp arquivo-origem caminho/diretório-destino/
4
5 # cp -R diretório-origem nome-destino
6
7 # cp -R diretório-origem caminho/diretório-destino/
```

Uma opção do comando **“cp”** muito útil em nosso dia-a-dia é a opção **“-p”,** que faz com que a cópia mantenha os “meta-dados” dos arquivos, ou seja, não modifica a data e hora de criação, seus donos e nem suas permissões. Utilizar como root:

```
1 # su - aluno
2 $ touch teste
3 $ ls -l
4 $ exit
5 # cd /home/aluno
6 # cp -p teste teste2
7 # cp teste teste3
8 # ls -l teste2 teste3
```

4.1.7 Mover ou renomear arquivos/diretórios

O comando **“mv”** serve tanto para renomear um arquivo quanto para movê-lo:

```
1 # mv arquivo caminho/diretório-destino/
2  # mv arquivo novo-nome
3  # mv diretório novo-nome
4  # mv diretório caminho/diretório-destino/
```

A movimentação de um arquivo é uma ação de cópia seguida de uma remoção.

Renomeando arquivo:

```
1 # mv teste teste4
```

Movendo arquivo:

```
1 # mv teste4 /tmp
```

Renomeando diretório:

```
1 # cd /srv
2 # mv Financeiro financeiro
```

Movendo diretório:

```
1 # mv financeiro /srv/Rh/
```

FHS, Hierarquia dos Diretórios

4.2 Introdução teórica

Quem já teve algum contato com o GNU/Linux, mesmo que superficial, deve ter percebido a presença de vários diretórios (pastas) no sistema. Entretanto, eles estão organizados seguindo o padrão "POSIX", com o qual você pode não estar muito familiarizado. Neste capítulo, vamos conhecer a organização, e explorar a estrutura de diretórios de um sistema GNU/Linux.

Desde que o GNU/Linux foi criado, muito se tem feito para seguir um padrão em relação à estrutura de diretórios. O primeiro esforço para padronização de sistemas de arquivos para o GNU/Linux foi o **"FSSTND - Filesystem Standard"**, lançado no ano de 1994.

Cada diretório do sistema tem seus respectivos arquivos que são armazenados conforme regras definidas pela **"FHS - Filesystem Hierarchy Standard" ou "Hierarquia Padrão do Sistema de Arquivos"**, que define que tipo de arquivo deve ser guardado em cada diretório. Isso é muito importante, pois o padrão ajuda a manter compatibilidade entre as distribuições existentes no mercado, permitindo que qualquer software escrito para o GNU/Linux seja executado em qualquer distribuição desenvolvida de acordo com os padrões "FHS".

Atualmente, o "FHS" está na sua versão 2.3, e é mantido pelo "Free Standard Group", uma organização sem fins lucrativos formada por grandes empresas como HP, IBM, RedHat e Dell.

É vital entender bem sobre a “FHS” para prova, é através dela que nós devemos fazer nossas atividades com o GNU/Linux em nosso dia-a-dia.

4.3 Estrutura de Diretórios GNU/Linux

A estrutura de diretórios também é conhecida como “Árvore de Diretórios” porque tem a forma de uma árvore. Mas, antes de estudarmos a estrutura de diretórios, temos que entender o que são diretórios.

Um diretório é o local onde os arquivos são guardados no sistema. O objetivo é organizar os diferentes arquivos e programas. Pense nos diretórios como sendo as gavetas de um armário. Cada gaveta guarda, normalmente, um tipo diferente de roupa, enquanto cada diretório guarda um certo tipo específico de arquivo.

O arquivo pode ser um texto, uma imagem, planilha, etc. Os arquivos devem ser identificados por nomes para que sejam localizados por quem deseja utilizá-los.

Um detalhe importante a ser observado é que o GNU/Linux segue o padrão “POSIX” que é “case sensitive”, isto é, ele diferencia letras maiúsculas e minúsculas nos arquivos e diretórios.

Sendo assim, um arquivo chamado **“Arquivo”** é diferente de um outro chamado **“ARQUIVO”** e diferente de um terceiro, chamado “arquivo”. Inteligente isso, não é?

A árvore de diretórios do GNU/Linux tem a seguinte estrutura:

/								
bin	*cdrom*	*etc*	*lib*	*mnt*	*proc*	*root*	*var*	*sys*
boot	*dev*	*home*	*media*	*opt*	*sbin*	*srv*	*tmp*	*usr*

Da estrutura mostrada acima, o "FHS" determina que um sistema GNU/Linux deve conter obrigatoriamente 14 diretórios, especificados a seguir:

4.3.1 Diretório /

```
1  # ls --color /
```

A opção –color do comando ls serve para deixar colorido a listagem, ex: azul -> diretório branco -> arquivo regular verde -> arquivo executável azul claro -> link simbólico vermelho -> arquivo compactado rosa -> imagem

Este é o principal diretório do GNU/Linux, e é representado por uma "/" (barra). É no diretório raiz que ficam todos os demais diretórios do sistema. Estes diretórios, que vamos conhecer agora, são chamados de "subdiretórios" pois estão dentro do diretório "/".

4.3.2 Diretório /bin

```
1  # ls /bin
```

O diretório "**/bin**" guarda os comandos essenciais para o funcionamento do sistema.

Esse é um diretório público, sendo assim, os comandos que estão nele podem ser utilizados por qualquer usuário do sistema. Entre os comandos, estão:

- /bin/ls;
- /bin/cp;
- /bin/mkdir;
- /bin/cat;

Qualquer usuário pode executar estes comandos:

```
# /bin/ls /boot/grub
$ /bin/ls /boot/grub
```

4.3.3 Diretório /boot

```
# ls /boot
```

No diretório **"/boot"** estão os arquivos estáticos necessários à inicialização do sistema, e o gerenciador de "boot". O gerenciador de "boot" é um programa que permite escolher e carregar o sistema operacional que será iniciado.

4.3.4 Diretório /dev

```
# ls /dev
```

No diretório **“/dev”** ficam todos os arquivos de dispositivos. O GNU/Linux faz a comunicação com os periféricos por meio de “links” especiais que ficam armazenados nesse diretório, facilitando assim o acesso aos mesmos.

Para verificar que seu mouse é reconhecido como um arquivo, tente olhar o conteúdo do arquivo /dev/input/mice:

```
1 # cat /dev/input/mice
```

Repare que os dados são binários e não é possível ler o arquivo com o comando cat. Caso seu terminal fique com caracteres estranhos utilize o comando **“reset”** para resetar o shell:

```
1 # reset
```

Para visualizar o conteúdo do arquivo /dev/input/mice execute o comando **“od”** que é utilizado para visualizar o conteúdo de um arquivo nos formatos: hexadecimal, octal, ASCII e nome dos caracteres. Este comando pode ser útil para um programador que deseja criar um programa conforme o movimento do mouse.

```
1 # od /dev/input/mice
```

Caso seu mouse não seja usb, execute:

```
1 # od /dev/psaux
```

Mova o mouse e observe sua saída.

Observe o conteúdo do seu HD:

```
1 # hexdump /dev/sda
```

O comando **hexdump** é utilizado para visualizar o conteúdo de um arquivo nos formatos: hexadecimal, octal, decimal, ASCII. Este comando pode ser útil para um programador que deseja criar um programa conforme o movimento do mouse.

4.3.5 Diretório /etc

```
1 # ls /etc
```

No diretório **"/etc"** estão os arquivos de configuração do sistema. Nesse diretório vamos encontrar vários arquivos de configuração, tais como: "scripts" de inicialização do sistema, tabela do sistema de arquivos, configuração padrão para "logins" dos usuários, etc.

```
1 # cat /etc/passwd
```

Vamos pegar uma linha de exemplo:

```
1 aluno:x:1000:1000:aluno:/home/aluno:/bin/bash
```

Vamos dividir esta linha em "campos", onde cada campo é separado por : (dois pontos), então:

Campo	Significado
aluno	Login do Usuário.
x	Aqui diz que a senha está no arquivo /etc/shadow. Se estivesse *, a conta estaria desabilitada, e se estivesse sem nada (::), a conta não teria senha.
1000	UID (User IDentification), o número de identificação do usuário.
1000	GID (Group IDentification), o número de identificação do grupo do usuário.
aluno	GECOS Comentários do usuário, como nome, telefone, etc
/home/aluno	O diretório HOME do usuário.
/bin/bash	Shell do usuário, ou seja, o programa que irá interpretar os comandos que o usuário executar

Vamos conhecer o arquivo /etc/shadow:

```
1 # more /etc/shadow
```

O comando **more** assim como o **cat** serve para ver o conteúdo de um arquivo que é, geralmente, texto. A diferença entre o "more" e o "cat" é que o "more" faz uma pausa a cada tela cheia exibindo uma mensagem --More–", dando uma oportunidade ao usuário ler a tela.

Aperte **enter** para ir para a próxima linha ou **espaço** para ir para a próxima página e para sair digite q.

Uma alternativa ao uso do comando **more** seria o uso do comando **less**, que implementa as mesmas funcionalidades que more e mais algumas, como a possibilidade de rolar a tela para cima e para o lado quando o texto ocupa mais de oitenta colunas. A utilização dos comandos **less** e **more** se faz de maneira semelhante.

```
1 # less /etc/shadow
```

Vamos pegar uma linha de exemplo:

```
1 aluno:$1$Tcnt$Eisi0J9Wh3fCEsz1:11983:0:99999:7:::
```

Este arquivo possui as senhas criptografadas dos usuários do sistema. Existe uma entada no arquivo para cada usuário do sistema com os seguintes campos:

Campo	Significado
aluno	Login do Usuário.
1Tcnt$Eisi0J9Wh3fCEsz1	A senha criptografada.
11983	O número de dias que existem de 01/01/1970 até a data da última modificação da senha.
0	Número de dias antes do sistema permitir uma nova modificação na senha.
99999	Número máximo de dias que o usuário pode ficar com uma mesma senha
7	Número de dias, antes da expiração da senha, quando o usuário é informado da necessidade de alterar a senha.
	Número de dias, após a expiração da senha, quando a conta passa a ser considerada inativa (o valor -1 significa que a conta fica inativa no mesmo dia da data de expiração).
	Número de dias que existem de 01/01/1970 até a expiração da senha (o valor -1 significa que não há data de expiração)
	Campo reservado para uso futuro.

Apenas o usuário root (administrador do sistema) tem permissão para acessar o arquivo /etc/shadow.

O comando **"pwconv"** é usado para criar o arquivo shadow a partir do arquivo /etc/-passwd , enquanto o comando **"pwunconv"** executa a operação inversa. Execute:

```
1 #  pwunconv
```

Verifique que não existe mais o arquivo /etc/shadow:

```
1 # cat /etc/shadow
```

Verifique que as senhas criptografadas estão agora no arquivo /etc/passwd através do comando getent:

```
# getent passwd
```

O comando **getent** obtém dados da base administrativa do sistema, seguindo a ordem de busca que está no arquivo /etc/nsswitch.conf:

```
# cat /etc/nsswitch.conf
# /etc/nsswitch.conf
#
# Example configuration of GNU Name Service Switch functionality.
# If you have the ‘glibc-doc-reference’ and ‘info’ packages
   installed, try:
# ‘info libc "Name Service Switch"’ for information about this file.

passwd:         compat
group:          compat
shadow:         compat

hosts:          files mdns4_minimal [NOTFOUND=return] dns mdns4
networks:       files

protocols:      db files
services:       db files
ethers:         db files
rpc:            db files

netgroup:       nis
```

Observe a linha do passwd, o “compat” significa compatibilidade com o sistema, ou seja, o arquivo /etc/passwd, mas os usuários e as senhas poderiam estar armazenados em uma outra localidade, por exemplo em um servidor LDAP e se você apenas executasse um **“cat /etc/passwd”**, não veria todos os usuários do sistema, então sempre utilize o **“getent passwd”** porque não importa onde os dados estão armazenados ele sempre seguirá a ordem de busca do arquivo /etc/nsswitch.conf.

Para voltar as senhas criptografadas, execute:

```
# pwconv
```

Agora as senhas estão protegidas novamente!! Antigamente estes comandos eram utilizados para sistemas que não vinham com as senhas protegidas no /etc/shadow por padrão, hoje em dia praticamente todas as distribuições trazem o arquivo como padrão, então utilizamos o comando para execução de scripts para facilitar a captura de senhas, como por exemplo a migração de um servidor de e-mail, onde queremos manter a senha antiga do usuário.

4.3.6 Diretório /lib

```
 # ls /lib
```

No diretório "**/lib**" estão as bibliotecas compartilhadas e módulos do kernel. As bibliotecas são funções que podem ser utilizadas por vários programas.

Cada kernel têm seus próprios módulos, que ficam em: /lib/modules/<versão do kernel>/kernel Separados por tipos em subdiretórios.

Para saber sua versão do kernel execute:

```
# uname -r
```

Para visualizar os tipos de módulos:

```
# ls /lib/modules/$(uname -r)/kernel
```

4.3.7 Diretório /media

```
1 # ls /media
```

Ponto de montagem para dispositivos removíveis, tais como:

- **hd**
- **cd**
- **dvd**
- **disquete**
- **pendrive**
- **câmera digital**

4.3.8 Diretório /mnt

```
1 # ls /mnt
```

Este diretório é utilizado para montagem temporária de sistemas de arquivos, tais como compartilhamentos de arquivos entre Windows e GNU/Linux, GNU/Linux e GNU/Linux, etc.

4.3.9 Diretório /opt

```
1  # ls /opt
```

Normalmente, é utilizado por programas proprietários ou que não fazem parte oficialmente da distribuição.

4.3.10 Diretório /sbin

```
1  # ls /sbin
```

O diretório **"/sbin"** guarda os comandos utilizados para inicializar, reparar, restaurar e/ou recuperar o sistema. Isso quer dizer que esse diretório também contém comandos essenciais, mas os mesmos são utilizados apenas pelo usuário administrador "root". Entre os comandos estão:

- **halt**
- **ifconfig**
- **init**
- **iptables**

Os usuários comuns não podem executar comandos do /sbin que alterem o sistema, apenas alguns para visualização.

EX:

Visualizar IP configurado na placa eth0:

- $ /sbin/ifconfig eth0

Alterar IP da placa de rede:

- $ /sbin/ifconfig eth0 192.168.200.100

Obs.: é necessário passar o caminho completo do comando, pois o diretório /sbin não consta na lista de diretórios de comandos do usuário comum que é definida na variável PATH, iremos estudar esta variável durante o curso.

4.3.11 Diretório /srv

```
1 # ls /srv
```

Diretório para dados de serviços fornecidos pelo sistema, cuja aplicação é de alcance geral, ou seja, os dados não são específicos de um usuário. Por exemplo:

- /srv/www (servidor web)

- /srv/ftp (servidor ftp)

4.3.12 Diretório /tmp

```
1 # ls /tmp
```

Diretório para armazenamento de arquivos temporários. É utilizado principalmente para guardar pequenas informações que precisam estar em algum lugar até que a operação seja completada, como é o caso de um “download”.

Enquanto não for concluído, o arquivo fica registrado em **"/tmp",** e, assim que é finalizado, é encaminhado para o local correto.

No Debian os dados são perdidos a cada reboot, já no CentOS os dados são mantidos durante dez dias após seu último acesso.

Para alterar no Debian:

```
# vim /etc/default/rcS
TMPTIME=0
SULOGIN=no
DELAYLOGIN=no
UTC=yes
VERBOSE=no
FSCKFIX=no
```

Altere o valor da variável "TMPTIME" para o número de dias que desejar manter os dados após o seu último acesso.

Para alterar no CentOS:

```
# vim /etc/cron.daily/tmpwatch
flags=-umc
/usr/sbin/tmpwatch "$flags" -x /tmp/.X11-unix -x /tmp/.XIM-unix \
  -x /tmp/.font-unix -x /tmp/.ICE-unix -x /tmp/.Test-unix \
  -X '/tmp/hsperfdata_*' 10d /tmp
/usr/sbin/tmpwatch "$flags" 30d /var/tmp
for d in /var/{cache/man,catman}/{cat?,X11R6/cat?,local/cat?}; do
    if [ -d "$d" ]; then
  /usr/sbin/tmpwatch "$flags" -f 30d "$d"
    fi
done
```

Altere de 10 dias, para o total de dias que quiser.

4.3.13 Diretório /usr

```
1 # ls /usr
```

O diretório **"/usr"** contém programas que não são essenciais ao sistema e que seguem o padrão GNU/Linux, como, por exemplo, navegadores, gerenciadores de janelas, etc.

Fique atento as diferenças entre: /bin - binários essenciais ao sistema /usr/bin - binários não essenciais ao sistema /usr/local/bin - scripts criados pelo usuário

4.3.14 Diretório /var

```
1 # ls /var
```

O diretório **"/var"** contém arquivos de dados variáveis. Por padrão, os programas que geram arquivos de registro para consulta, mais conhecidos como "logs", ficam armazenados nesse diretório. Além do "log", os arquivos que estão aguardando em filas, também ficam localizados em "/var/spool".

Os principais arquivos que se utilizam do diretório "/var" são:

- mensagens de e-mail
- arquivos a serem impressos

```
1 # ls /var/spool
```

arquivos de log

```
1 # ls /var/log
```

4.3.15 Diretório /proc

```
1 # ls /proc
```

O **"/proc"** é um diretório virtual, mantido pelo **kernel,** onde encontramos a configuração atual do sistema, dados estatísticos, dispositivos já montados, interrupções, endereços e estados das portas físicas, dados sobre as redes, etc.

Utilize os paginadores more ou less para visualizar alguns arquivos:

```
1 # more /proc/interrupts
```

Neste arquivo estão as informações das **IRQs dos dispositivos.**

Os endereços de IRQ são interrupções de hardware, canais que os dispositivos podem utilizar para chamar a atenção do processador.

Na maioria das situações, o sistema operacional simplesmente chaveia entre os aplicativos ativos, permitindo que ele utilize o processador durante um determinado espaço de tempo e passe a bola para o seguinte. Como o processador trabalha a uma frequência de clock muito alta, o chaveamento é feito de forma muito rápida, dando a impressão de que todos realmente estão sendo executados ao mesmo tempo.

Ao ser avisado através de qualquer um destes canais de IRQ, o processador imediatamente pára qualquer coisa que esteja fazendo e dá atenção ao dispositivo, voltando

ao trabalho logo depois. Cada endereço é uma espécie de campainha, que pode ser tocada a qualquer momento. Se não fossem pelos endereços de IRQ, o processador não seria capaz de ler as teclas digitadas no teclado, nem os clicks do mouse, a sua conexão pararia toda vez que abrisse qualquer programa e assim por diante.

```
1 # less /proc/dma
```

É o arquivo que contém a lista do registro ISA direto dos canais em uso da acesso a memória (DMA).

Os canais de **DMA** são utilizados apenas por dispositivos de legado (placas ISA, portas paralelas e drives de disquete) para transferir dados diretamente para a memória RAM, reduzindo desta forma a utilização do processador.

```
1 # more /proc/ioports
```

Neste arquivo encontramos informações sobre os endereços das portas I/O (Input/Output).

Diferentemente dos endereços de IRQ, os endereços de I/O não são interrupções, mas sim endereços utilizados para a comunicação entre os dispositivos. Cada dispositivo precisa de um endereço próprio mas, ao contrário dos endereços de IRQ, existe uma abundância de endereços de I/O disponíveis, de forma que eles raramente são um problema.

4.3.16 Diretório /sys

```
1 # ls /sys
```

Pode-se dizer que esse diretório é um primo do diretório **"/proc".** Dentro do diretório **"/sys"** podemos encontrar o quase o mesmo conteúdo do "/proc", mas de uma forma bem mais organizada para nós administradores.

Esse diretório está presente desde a versão 2.6 do kernel, ele agrupa informações sobre os dispositivos instalados, incluindo o tipo, fabricante, capacidade, endereços usados e assim por diante. Estas informações são geradas automaticamente pelo kernel e permitem que os serviços responsáveis pela detecção de hardware façam seu trabalho, configurando impressoras e criando ícones no desktop para acesso ao pendrive, por exemplo.

4.3.17 Diretórios /home e /root

```
1 # ls /home  /root
```

Os diretórios **"/root" e "/home"** podem estar disponíveis no sistema, mas não precisam obrigatoriamente possuir este nome. Por exemplo, o diretório "/home" poderia se chamar "/casa", que não causaria nenhum impacto na estrutura do sistema.

O **"/home"** contém os diretórios pessoais dos usuários cadastrados no sistema.

O **"/root"** é o diretório pessoal do super usuário "root".

O "root" é o administrador do sistema, e pode alterar as configurações do sistema, configurar interfaces de rede, manipular usuários e grupos, alterar a prioridade dos processos, entre outras. Dica: Utilize uma conta de usuário normal em vez da conta "root" para operar seu sistema.

Uma razão para evitar usar privilégios "root" regularmente, é a facilidade de se cometer danos irreparáveis; além do que, você pode ser enganado e rodar um

programa “Cavalo de Tróia” (programa que obtém poderes do super usuário) comprometendo a segurança do seu sistema sem que você saiba.

4.4 Localização no sistema:

4.4.1 Find

O comando “find” procura por arquivos/diretórios no disco. Ele pode procurar arquivos pela sua data de modificação, tamanho, etc. O “find”, ao contrário de outros programas, usa opções longas por meio de um ”-”.

Sintaxe do comando **“find”:**

 find [diretório] [opções/expressão]

- **-name [expressão] :**

Procura pela [expressão] definida nos nomes de arquivos e diretórios processados.

```
1 # find /etc -name *.conf
```

- **-maxdepth [num] :**

Limita a recursividade de busca na árvore de diretórios. Por exemplo, limitando a 1, a busca será feita apenas no diretório especificado e não irá incluir nenhum subdiretório.

```
1 # find /etc -maxdepth 1 -name *.conf
```

- **-amin [num] :**

Procura por arquivos que foram acessados [num] minutos atrás. Caso seja antecedido por "-", procura por arquivos que foram acessados entre [num] minutos atrás e o momento atual.

```
1 # find ~ -amin -5
```

- **-atime [num] :**

Procura por arquivos que foram acessados [num] dias atrás. Caso seja antecedido por "-", procura por arquivos que foram acessados entre [num] dias atrás e a data atual.

```
1 # find ~ -atime -10
```

- **-uid [num] :**

Procura por arquivos que pertençam ao usuário com o "uid 1000" [num].

```
1 # find / -uid 1000
```

- **-user [nome] :**

Procura por arquivos que pertençam ao usuário "aluno" [nome].

```
1  # find / -user aluno
```

- **-perm [modo] :**

Procura por arquivos que possuem os modos de permissão [modo]. Os [modo] de permissão podem ser numérico (octal) ou literal.

```
1  # find / -perm 644
```

- **-size [num] :**

Procura por arquivos que tenham o tamanho [num]. O tamanho é especificado em bytes. Você pode usar os sufixos k, M ou G para representar o tamanho em Quilobytes, Megabytes ou Gigabytes, respectivamente. O valor de [num] Pode ser antecedido de "+" ou "-" para especificar um arquivo maior ou menor que [num].

```
1  # find / -size +1M
```

-type [tipo] :

Procura por arquivos do [tipo] especificado. Os seguintes tipos são aceitos:

b - bloco; c - caractere; d - diretório; p - pipe; f - arquivo regular; l - "link" simbólico; s - "socket".

```
1  # find /dev -type b
```

Outros exemplos:

Procura no diretório raiz e nos subdiretórios um arquivo/diretório chamado “grep” ignorando caso sensitivo:

```
1  # find / -iname GREP
```

-iname - ignora case sensitive;

Procura no diretório raiz e nos subdiretórios até o 2º nível, um arquivo/diretório chamado “grep”:

```
1 # find / -maxdepth 2 -name grep
```

Procura no diretório atual e nos subdiretórios um arquivo com tamanho maior que 1000 kbytes (1Mbyte).:

```
1  # find . -size +1000k
```

Procura no diretório raiz e nos subdiretórios um arquivo que foi modificado há 10 minutos atrás ou menos:

```
1  # find / -mmin -10
```

Procura diretórios a partir do diretório /etc e também executa um comando no resultado da busca com a opção “exec”, no exemplo o comando é “ls -ld”:

```
1 # find /etc -type d  -exec ls -ld {} \;
```

Xargs

Outra forma de procurar por arquivos e/ou diretórios e executar um comando é através do comando xargs que obtém como a entrada a saída ok do comando antes do pipe e envia como stdin do próximo comando, no caso o ls -ld:

```
1 # find /etc -type d  | xargs ls -ld
```

Vamos agora listar diretórios utilizando o “xargs”:

```
1 # ls / | xargs -n1
2 # ls / | xargs -n2
3 # ls / | xargs -n3
```

Outros testes com o “xargs”:

```
1  # ls / > teste_xargs.txt
2  # cat teste_xargs.txt
3  # cat teste_xargs.txt | xargs -n 2
4  # xargs -n 3 < teste_xargs.txt
```

Você percebeu que no primeiro comando ele listou o diretório, jogando na tela um nome de cada vez. O segundo comando fará o mesmo só que com dois nomes na mesma linha, e o terceiro com 3 nomes.

Tempo de execução de um programa: time

O comando **“time”** permite medir o tempo de execução de um programa. Sua sintaxe é: time [programa].

Exemplo:

```
1  # time find / -name *.conf
```

4.4.2 Locate

O comando “locate” é um comando rápido de busca de arquivos, porém não usa busca recursiva na sua árvore de diretórios. Ele utiliza uma base de dados que é criada pelo comando “updatedb”, para que a busca seja mais rápida. Por padrão, a atualização da base de dados é agendado no “cron” do sistema para ser executada diariamente.

Para utilizá-lo, primeiro é necessário criar a sua base de dados usando a seguinte sintaxe:

```
1  # updatedb
```

Quando esse comando é executado pela primeira vez costuma demorar um pouco. Isso deve-se a primeira varredura do disco para a criação da primeira base de dados. Para o comando **“locate”,** usamos a seguinte sintaxe:

```
1  # locate howto
```

A saída do comando será algo parecido com:

```
1  /usr/share/doc/python2.4-xml/howto.cls
2  /usr/share/doc/python2.4-xml/xml-howto.tex.gz
3  /usr/share/doc/python2.4-xml/xml-howto.txt.gz /usr/share/vim/vim64/
      doc/howto.txt
```

455

Linux Essentials

www.4linux.com.br

Conteúdo

Comandos Avançados I

5.1 Introdução teórica

No mundo GNU/Linux, a maioria das operações são realizadas por meio de comandos escritos. Em geral, eles permitem um maior controle e flexibilidade de operações, além de poderem ser incluídos em "scripts". Neste capítulo iremos aprender alguns comandos avançados.

5.1.1 Trabalhando com entrada e saída de dados

No linux, você pode ler dados de um arquivo ou terminal ou escrever dados para um arquivo ou terminal. O linux tem três tipos de fluxo de dados: entrada(INPUT), saída(OUTPUT) e a última para imprimir diagnósticos ou mensagens de erro.

Por padrão, a entrada de dados e comandos no "Shell" é feita pelo teclado, a saída destes é retornada na tela. Eventuais erros são exibidos na tela também. Porém você pode alterar a saída padrão que é a tela e enviá-la para um arquivo ou outra localização.

Os termos geralmente usados são: 0 - Entrada de dados, representada por **"stdin"** (Standard Input); 1 - Saída de dados, representada por **"stdout"** (Standard Output); 2 - Saída de erros, representada por **"stderr"** (Standard Error);

5.1.2 Alterando os redirecionamentos

Formas de redirecionar o fluxo de dados:

> (maior): Direciona a saída do comando para um arquivo, substituindo o seu conteúdo, caso o arquivo já exista;

» (maior-maior): Direciona a saída do comando para um arquivo, adicionando o texto ao final do arquivo, caso ele já exista;

< (menor): Passa o conteúdo do arquivo como argumento para o comando;

2> (dois-maior): Direciona as saídas de erro geradas pelo programa para um arquivo, substituindo seu conteúdo, caso o arquivo já exista;

2» (dois-maior-maior): Direciona as saídas de erro geradas pelo programa para um arquivo, adicionando o texto ao final do arquivo, caso ele já exista;

2>&1 (dois-maior-e-um): Direciona as saídas de erro para a saída do comando, no caso para STDOUT;

&> (e-maior): Direciona todas as saídas (normal e de erro) para um arquivo, substituindo seu conteúdo, caso ele já exista; &» (e-maior-maior): Direciona todas as saídas (normal e de erro) para um arquivo, adicionando o texto ao final do arquivo, caso ele já exista;

| (barra vertical ou pipe): Utiliza a saída do primeiro comando como argumento do segundo comando; tee: mostra saída na tela e redireciona para um arquivo ou outra localização ao mesmo tempo;

«: marca o fim de um bloco.

O direcionador >

O direcionador > direciona a saída padrão de um comando para um arquivo. Caso o arquivo exista, o seu conteúdo é substituído.

Vejamos, então, uma saída do comando ls:

```
1 # ls /
2 boot   lost+found   mnt      repo     srv  var      cdrom     etc
        lib    media       opt      root     sys   home   sbin
    tmp  bin   dev   lib64   proc     selinux  usr
```

Para gravar essa lista em um arquivo chamado raiz, utilizamos o direcionador, da seguinte forma:

```
1 # ls / > raiz
```

Não aparece nada na tela porque o comando foi executado sem erros e sua saída redirecionada para o arquivo raiz, confira:

```
1 # cat raiz
```

O conteúdo do arquivo raiz é o mesmo da saída do comando ls. Cuidado ao utilizar o direcionador para o mesmo arquivo, pois os dados serão perdidos, exemplo:

Quero enviar a saída do arquivo raiz para raiz:

1 - Primeiro visualize o arquivo para ver que há dados no arquivo raiz:

```
1 # cat raiz
```

2 - envie a saída do cat para o arquivo raiz:

```
1 # cat raiz > raiz
```

Ao realizar o comando acima, a primeira interpretação do bash é executar o comando: **“> raiz”**, ou seja, se não existe o arquivo, ele será criado, e se já existe é sobrescrito. No caso ele sobrescreve o arquivo raiz, deixando-o em branco, e quando o comando **“cat raiz”** é executado, não há saída, pois o arquivo está zerado, não redirecionando nada.

Para evitar este problema execute o comando:

```
1 # set -o noclobber
```

após o comando faca o exemplo:

```
1 cat /etc/fstab > hoje
```

```
1 cat hoje > hoje
```

Verifique que o arquivo nao foi sobrescrito.

e para voltar:

```
1 # set +o noclobber
```

5.1.3 O direcionador »

O direcionador » direciona a saída padrão de um comando para um arquivo. Caso o arquivo exista, a saída é adicionada ao final do arquivo.

```
# ls / >> hoje
```

Verifique que a saída do comando ls foi adicionada ao final do arquivo hoje.

5.1.4 O direcionador <

O direcionador < é utilizado para passar um stdin para um comando, ele é geralmente utilizado para passar o conteúdo de arquivos como parâmetros de comandos.

Alguns comandos precisam que seja passado o stdin para eles serem executados, vamos ver o exemplo do comando **tr**, que traduz ou deleta caracteres:

Para converter letras minúsculas por maiúsculas faça:

```
# tr "a-z" "A-Z" /etc/passwd
```

Verifique que sem o redirecionador < o comando não é executado com sucesso, agora faça corretamente:

```
# tr "a-z" "A-Z" < /etc/passwd
```

Você também pode utilizar o comando tr para deletar caracteres, vamos deletar as vogais do arquivo:

```
1 # tr -d aeiou < /etc/passwd
```

Para que as mudanças sejam efetuadas de fato é necessário encaminhar a saída para outro arquivo.

5.1.5 O direcionador 2>

Quando utilizamos o direcionador > ele não redireciona as saída de erro, apenas a saída sem erros, caso o arquivo não exista será criado e caso já exista será sobrescrito. Por exemplo, vamos usar o comando ls usando como parâmetro um diretório que não existe e redirecionar sua saída para um novo arquivo:

```
1 # ls nao_existe > ls_naoexiste
2
3 ls: impossível acessar nao_existe: Arquivo ou diretório não
     encontrado
```

Verifique que mesmo não redirecionando a saída com erro o arquivo ls_naoexiste é criado:

```
1 # cat ls_naoexiste
```

Para gravar as mensagens de erro, devemos utilizar o direcionador 2>:

```
1 # ls nao_existe 2> ls_naoexiste.err
```

Agora sim, nenhuma mensagem de erro foi exibida na tela, porque ela foi enviada para o arquivo ls_naoexiste.err, vamos verificar o conteúdo dele:

```
1 # cat ls_naoexiste.err
2 ls: impossível acessar nao_existe: Arquivo ou diretório não
     encontrado
```

5.1.6 O direcionador 2»

Quando utilizamos o direcionador 2» ele redireciona apenas as mensagens de erro, caso o arquivo não exista será criado e caso já exista será adicionada a saída ao final do arquivo.

```
1 # cat ls_naoexiste.err
```

Agora vamos redirecionar outra saída de erro para este arquivo:

```
1 # cat /nada 2>> ls_naoexiste.err
```

Verifique que a saída de erro foi adicionada ao arquivo ls_naoexiste.err:

```
1 # cat  ls_naoexiste.err
2 ls: impossível acessar nao_existe: Arquivo ou diretório não
     encontrado
3 cat: /nada: Arquivo ou diretório não encontrado
```

5.1.7 O direcionador 2>&1

Podemos usar os direcionadores > e 2> em conjunto, para gerar um arquivo com a saída padrão e outro com a saída de erros, dessa forma:

```
1 # cat /etc/*
```

A saída mostra tanto o conteúdo dos arquivos quanto os erros por tentar ler um diretório com o comando cat.

Vamos enviar a saída deste comando para arquivos diferentes:

```
1 # cat /etc/* > msg_correto 2> msg_errado
```

Visualize o conteúdo dos arquivos msg_correto e msg_errado:

```
1 # cat msg_correto
2 # cat msg_errado
```

Mas, e se for necessário gravar todas as mensagens em um arquivo apenas?

Podemos redirecionar o stderr para o stdout:

```
1 # cat /etc/* > msg_total 2>&1
```

Aqui redirecionamos o stdout para o arquivo msg_total e redirecionamos o stderr para stdout, ou seja, também para o arquivo msg_total.

Visualize seu conteúdo:

```
1 # cat  msg_total
```

5.1.8 O direcionador >

Podemos usar os direcionadores > e 2> em conjunto, para gerar um arquivo com a saída padrão e outro com a saída de erros, dessa forma:

```
1 # cat /etc/*
```

A saída mostra tanto o conteúdo dos arquivos quanto os erros por tentar ler um diretório com o comando cat.

Vamos enviar a saída deste comando para arquivos diferentes:

```
1 # cat /etc/* > msg_ok 2> msg_error
```

Visualize o conteúdo dos arquivos msg_ok e msg_error:

```
1 # cat msg_ok
2 # cat msg_error
```

Mas, e se for necessário gravar todas as mensagens em um arquivo apenas? Para isso existe o direcionador >, que direciona tanto as mensagens padrão quanto as mensagens de erro para um único arquivo, caso o arquivo não exista será criado e caso já exista será sobrescrito.

Repetindo o teste anterior:

```
1 # cat /etc/* &> ls_out
```

Não aparece nenhuma mensagem no terminal, pois tanto as mensagens ok quanto as mensagens com erro foram redirecionadas para o arquivo ls_out, visualize seu conteúdo:

```
1 # cat ls_out
```

5.1.9 O direcionador &»

Assim como o redirecionador &> ele redireciona tanto a saída de stdout quanto a saída de stderr para um único arquivo, a diferença é que, caso o arquivo não exista ele será criado e caso já exista será adicionado a saída com comando ao final do arquivo. Visualize o arquivo ls_out:

```
1 # cat ls_out
```

Agora redirecione a saída stdout e stderr para ele com &» :

```
1 # cat /etc/* &>> ls_out
```

Não aparece nenhuma mensagem no terminal, pois tanto as mensagens ok quanto as mensagens com erro foram redirecionadas para o arquivo ls_out, visualize seu conteúdo:

```
1 # cat ls_out
```

Observe que a saída foi adicionada ao final do arquivo.

5.1.10 O direcionador |

Conhecido como pipe, ele envia o stdout de um comando para o stdin do próximo comando para dar continuidade ao processamento, os dados enviados serão processados pelo próximo comando trazendo assim um resultado esperado.

Vamos usar novamente o comando **tr** para exemplificar, mas desta vez utilizando o pipe:

Primeiro visualize o conteúdo do arquivo /etc/passwd:

```
1 # cat /etc/passwd
```

A saída foi o stdout do comando. Vamos agora redirecionar este stdout para o comando **"tr":**

```
1 # cat /etc/passwd | tr "a-z" "A-Z"
```

5.1.11 O direcionador tee

Quando usado junto com o pipe |, o tee permite que a saída padrão do comando seja exibida na tela e enviada para um arquivo ao mesmo tempo. Veja a saída de um comando e envie-a para um arquivo qualquer, caso o arquivo não exista, será criado e caso já exista será sobrescrito, caso queira adicionar à um arquivo já existente use **"tee -a" :**

```
1 # cat /etc/fstab | tee arquivo.tee
```

A saída aparece na tela e também foi direcionada para o arquivo “arquivo.tee”, visualize-o:

```
# cat arquivo.tee
```

5.1.12 O direcionador «

Temos ainda o direcionador «, utilizado para marcar o fim de exibição de um bloco. Um dos usos mais freqüentes desse direcionador é em conjunto com o comando cat.

Você pode editar um novo arquivo com o comando cat ou até mesmo adicionar conteúdo nele, veja:

```
# cat << EOF > arquivo_novo
```

Onde: « EOF - indica que a edição do arquivo terminará quando em uma linha contiver apenas a sequência **EOF.**

> arquivo_novo - direciona o que for digitado no arquivo para arquivo_novo. Ex:

```
# cat << EOF > arquivo_novo

Este
é
meu arquivo!
EOF
```

Visualize o arquivo gerado:

```
1 # cat arquivo_novo
```

5.1.13 Comando dd

O comando **dd** dos sistemas baseados em Linux e Unix, é um programa para copiar e converter arquivos de um local para outro, servindo até para clonar dispositivos ou áreas de discos fixos ou removíveis como CD(s), DVD(s), disquetes, HD(s), dispositivos USB entre outros.

Sintaxe:

```
1 # dd if=<origem> of=<destino>
```

Criando um arquivo de 1MB:

```
1 # dd if=/dev/zero of=teste.txt bs=1024 count=1000
```

Onde:

If - Input File = arquivo de origem

of - Output File = arquivo de destino

bs - Block Size = tamanho do bloco

count - número de blocos

Em sistemas operacionais do tipo Unix, /**dev**/**zero** é um arquivo especial que fornece quantos caracteres nulos (o NULL da tabela ASCII, 0x00; e não o caractere "dígito

zero", "0", 0x30) forem lidos dele. O fluxo de caracteres nulos gerado por este dispositivo pode, por exemplo, ser utilizado para sobreescrever informações ou para gerar um arquivo limpo de certo tamanho.

5.1.14 Comando wc

Grande parte dos arquivos de configuração e de dados usa uma linha por registro. A contagem destas linhas pode nos fornecer informações muito interessantes.

Por exemplo, a saída abaixo:

```
1  # wc /etc/passwd
```

Indica que o arquivo contém X linhas, Y blocos (palavras) e Z caracteres. Caso seja necessário apenas o número de linhas, o comando **“wc”** pode ser usado com o parâmetro “-l”, como abaixo:

```
1  # wc -l /etc/passwd
```

Apenas contar o número de blocos (palavras):

```
1 # wc -w /etc/passwd
```

Apenas contar o número de caracteres:

```
1 # wc -c /etc/passwd
```

5.1.15 Comando split

O comando split é usado para dividir determinado arquivo em pedaços menores, muito útil quando se tem um arquivo maior do que um espaço de armazenamento como por exemplo um cd, você pode dividir o arquivo para que ele caiba em dois ou mais cds, por exemplo.

Vamos dividir o arquivo gerado pelo dd em partes de 300Kb:

```
1 # split -b 300KB teste.txt
```

O tamanho pode ter os sufixos:

b 512, **KB** 1000, **K** 1024, **MB** 1000*1000, **M** 1024*1024

GB 1000*1000*1000, **G** 1024*1024*1024

Verifique que foram gerados vários arquivos com o prefixo “x”, veja também o tamanho deles:

```
1 # ls -lh xx*
```

Podemos dividir o arquivo por número de linhas e mudar seu prefixo também:

```
1 # split -l 10 /etc/passwd pref
```

Verifique que foram gerados vários arquivos com o prefixo “pref”, veja também o número de linhas de cada um:

```
1 # wc -l pref*
```

Para recuperar o arquivo, concatene todos os pedaços na ordem correta:

```
# cat  prefaa prefab prefac prefad > passwd.backup
```

Visualize o arquivo:

```
 # cat passwd.backup
```

5.1.16 Comando file

Extensões de arquivos têm apenas a função de nos auxiliar a nomear os arquivos, a identificá-los e organizá-los facilmente. Não é a extensão que determina o tipo do arquivo, mas sim o seu conteúdo. Por exemplo, se renomearmos um arquivo de imagem chamado 4Linux.jpg para 4Linux.html, ele continuará sendo um arquivo de imagem **"JPEG".**

O comando "file" determina o tipo do arquivo analisando o seu conteúdo. O exemplo abaixo mostra o uso deste comando:

```
# file /
# file /bin/cat
# file /dev/sda1
# file /etc/passwd
# file /usr/sbin/adduser
```

5.1.17 Comando who

Determina quais usuários estão logados.

Verifique os usuários que estão logados no sistema:

```
1 # who
2 fabiano  tty7         2011-08-11 23:01 (:0)
3 fabiano  pts/0        2011-08-12 21:32 (:0.0)
```

Onde:

fabiano - usuário logado tty7 - terminal em que o usuário está logado 2011-08-11 23:01 - hora e data de login (:0) - Display

5.1.18 Comando w

Mostra tempo que o sistema está ligado, média de carga do sistema, usuários logados.

```
1 # w
2 21:43:12 up 22:42,  2 users,  load average: 0,55, 0,45, 0,37
3 USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU  WHAT
4 fabiano  tty7     :0                Thu23   22:41m  6:27   0.53s
     gnome-session
5 fabiano  pts/0    :0.0              21:32    0.00s  0.22s  0.02s w
```

5.1.19 Comando ln

O comando **ln** permite criar links. Existem dois tipos de links suportados pelo Linux, os hard links e os links simbólicos. Os links simbólicos têm uma função parecida com os atalhos do Windows: eles apontam para um arquivo, mas se o arquivo é movido para outro diretório, o link fica quebrado. Os hard links por sua vez são

semelhantes aos atalhos do OS/2 da IBM, eles são mais intimamente ligados ao arquivo e são alterados junto com ele. Se o arquivo muda de lugar, o link é automaticamente atualizado. Isto é possível porque nos sistemas de arquivos usados pelo Linux cada arquivo possui um código de identificação (chamado de inode), que nunca muda. O sistema sabe que o arquivo renomeado é o mesmo do atalho simplesmente procurando-o pelo inode ao invés do nome.

5.1.20 Inodes

Cada diretório e arquivo do Linux é identificado para o kernel como um número de nó i (inode). Um inode é, na realidade, uma estrutura de dados que possui informações sobre um determinado arquivo ou diretório como, por exemplo, dono, grupo, tipo e permissão de acesso.

O inode é exclusivo somente para o dispositivo (partição) dentro do qual ele está contido. Portanto, para identificar unicamente um arquivo, o kernel deve ter o número de dispositivo e o inode do arquivo.

Um arquivo possui um único inode, não importa por quantos nomes este arquivo é identificado no sistema. Logo, é o conjunto de inodes que indica o número de arquivos/diretórios que o sistema possui.

5.1.21 Comando stat

Para saber o número do inode de um arquivo digite:

```
1 # ls -i <arquivo>
```

Ou utilize o comando stat:

```
1 # stat <arquivo>
```

Vamos ver o número do inode do arquivo /etc/passwd:

```
1 # stat /etc/passwd
2 1     File: '/etc/passwd'
3 2     Size: 3020  Blocks: 8   IO Block: 4096   arquivo comum
4 3   Device: 801h/2049d  Inode: 4995196     Links: 1
5 4     Access: (0644/-rw-r--r--)  Uid: (0/root)   Gid: (0/root)
6 5     Access: 2011-08-13 23:20:01.046079196 -0300
7 6   Modify: 2011-08-13 23:20:00.236496119 -0300
8 7   Change: 2011-08-13 23:20:00.325887191 -0300
```

Onde:

linha 1 - nome do arquivo.

linha 2 - tamanho, tipo do arquivo.

linha 3 - localização no dispositivo, **número do inode.**

linha 4 - permissões, dono, grupo.

linha 5 - última vez que o arquivo foi acessado, visualizado.

linha 6 - última vez que foi modificado o arquivo.

linha 7 - última vez que foi alterada a permissão do arquivo.

5.1.22 Link simbólico

É possível criar links simbólicos de arquivos e/ou diretórios mesmo que estejam em partições diferentes, já que o número de inodes do arquivo original e do link simbólico são diferentes, mas se o arquivo original for apagado o link é quebrado, tornando-se inútil. Também não é possível determinar a permissão olhando o link simbólico, somente olhando o original.

Vamos criar um arquivo para testarmos:

```
1  # vim arquivo
2  Este arquivo é para teste!
```

Agora vamos olhar o número do inode do arquivo com o comando ls:

```
1 # ls -i arquivo
2 11927685 arquivo
```

Onde:

-i - mostra número de inode do arquivo/diretório

Criando o link simbólico:

```
1 # ln -s <arquivo_original> <link_simbolico>
```

Vamos criar o link do arquivo: “arquivo” para o arquivo: “arq.simbolico”.

```
1 # ln -s arquivo arq.simbolico
```

Visualize os números de inodes e a permissão:

```
# ls -1il arq*
11927715 lrwxrwxrwx 1 root root 7 2011-08-11 18:04 arq.simbólico ->
    arquivo
11927685 -rw-r--r-- 1 root root 0 2011-08-11 17:52 arquivo
```

Onde:

-1 - mostra um arquivo por linha -i - mostra número do inode do arquivo/diretório -l - modo estendido

Verifique que não é possível determinar qual é a permissão olhando o arq.simbólico.

Crie um diretório:

```
# mkdir diretorio
```

Agora vamos olhar o número do inode do diretório com o comando ls:

```
# ls -di diretorio
11935762 diretorio
```

Onde:

-d - mostra informações do diretório -i - mostra número de inode do arquivo/diretório

Criando o link simbólico:

```
# ln -s <diretorio_original> <diretorio_simbolico>
```

Vamos criar o link do arquivo: “diretório” para o diretório: “dir.simbólico”.

```
1 # ln -s diretorio dir.simbólico
```

Visualize os números de inodes:

```
1 # ls  -1dil dir*
2 11935762 drwxr-xr-x 2 root root 4096 2011-08-11 18:12 diretório
3 11927717 lrwxrwxrwx 1 root root   10 2011-08-11 18:14 dir.simbólico
    -> diretório/
```

Onde:

-1 - mostra um arquivo/diretorio por linha -d - mostra informações do diretorio -i - mostra número do inode do arquivo/diretório -l - modo estendido

5.1.23 Hard links

Não é possível criar Hark links de arquivos e/ou diretórios que estejam em partições diferentes, pois o range de numeros de inodes mudam de uma partição para outra, ou seja, os Hark links não terão o mesmo número de inode, e também não é possível criar Hard links de diretórios da mesma partição.

Criando o Hard link:

```
1 # ln  <arquivo_original> <Hard_link>
```

Vamos criar o link do arquivo: “arquivo” para o arquivo: “arq.hard”.

```
1 # ln  arquivo arq.hard
```

Visualize os números de inodes:

```
1 # ls -1 -i arq*
```

Onde:

-1 - mostra um arquivo por linha -i - mostra número do inode do arquivo/diretório

5.1.24 Comando nl

O comando cat permite numerar as linhas através da opção “-n”:

```
1 # cat -n /etc/fstab
```

Existe um outro comando que também visualiza arquivo e numera as linhas, este Comando é o “nl”:

```
1 # nl /etc/passwd
2 # grep sys /etc/passwd | nl
3 # ls -l /etc | nl
4 # ls -l /etc | tail | nl
```

5.1.25 Comando sort

Para diversas ações como eliminação de itens repetidos e rápida visualização de nomes é interessante que possamos classificar um arquivo texto ou a saída de um comando. Na linha de comando, os arquivos textos podem ser classificados usando o comando “sort”.

Vamos criar um arquivo de exemplo:

```
1 # vim bagunça
2 Gabriela
3 Barbara
4 Bruno
5 Victor
6 Alexandre
7 Bruno
8 Alfredo
9 Bruno
```

A saída do comando abaixo não segue a ordem alfabética:

```
1 # cat bagunça
```

Podemos mostrar a saída classificada em ordem alfabética, assim:

```
1 # sort bagunça
```

O comando “sort” pode ser modificado usando os parâmetros:

-f - não considera se as letras estão em caixa alta ou baixa;

-n - classificação numérica;

-r - classifica na ordem invertida.

Para saber mais parâmetros:

```
1 # man sort
```

5.1.26 Comando uniq

Remove linhas desnecessárias ou duplicadas, ou seja, ele faz uma espécie de listagem de cada linha única do arquivo; Somente remove se as linhas repetidas estiverem na sequência, ou seja, uma após a outra, então sempre utilize o comando sort antes para ordenar as linhas.

Eliminando as linhas repetidas:

```
1 # sort bagunça | uniq
```

Para mostrar apenas as linhas que se repetem:

```
1 # sort bagunça | uniq -d
```

455

Linux Essentials

www.4linux.com.br

Conteúdo

Comandos Avançados II

6.1 Introdução teórica

No mundo GNU/Linux, a maioria das operações são realizadas por meio de comandos escritos. Em geral, eles permitem um maior controle e flexibilidade de operações, além de poderem ser incluídos em "scripts". Neste capítulo iremos aprender alguns comandos avançados.

6.1.1 Filtragem : grep e egrep e fgrep

6.1.2 Comando grep

Uma necessidade constante dos administradores é encontrar informações dentro dos arquivos. Para ilustrar, podemos localizar a palavra "root" no arquivo "/etc/passwd":

```
1   # grep root /etc/passwd
2  root:x:0:0:root:/root:/bin/bash
```

Outra situação possível é procurar pelas entradas que não possuem a palavra procurada. Para isso, usamos o parâmetro "-v" (inVerter), que inverte a filtragem do comando **"grep":**

```
# grep -v bash /etc/passwd
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
```

-v - Inverte a busca, encontra apenas as linhas onde o padrão não existir.

Traz como resultado todas as linhas do arquivo /etc/passwd, exceto as linhas que continham a palavra bash.

Para buscar a palavra "Debian" no arquivo "/etc/passwd" utilize a opção "-i" para ignorar maiúsculas e minúsculas:

```
# grep -in debian /etc/passwd
60:Debian-exim:x:123:132::/var/spool/exim4:/bin/false
```

-i - Ignora diferença entre maiúsculas e minúsculas; **-n -** Mostra o número de cada linha encontrada;

Caso queira ver além da linha que casar com a busca as duas próximas linhas e uma linha anterior, digite:

```
# grep -i debian /etc/passwd -A 2 -B 1
```

-A [n] - After = Mostra n linhas depois; **-B** [n] - Before = Mostra n linhas antes;

O "grep" pode ser combinado com a saída de outros comandos com o uso do "|" (pipe).

A seguir, o “grep” é aplicado para filtrar quem está logado no primeiro terminal (tty1):

```
1 # who |grep tty1
2 root     tty1         2011-08-14 22:40
```

6.1.3 Comando egrep

Para uma busca mais avançada utilize o “egrep”. Por exemplo quero buscar por uma linha que contenha a palavra “root” ou “aluno”:

```
1 # egrep "root|aluno" /etc/passwd
```

Procurar por linhas que contenham a palavra Debian ou debian:

```
1 # egrep [dD]ebian /etc/passwd
```

Procurar por linhas que começam com a letra “b”:

```
1 # egrep ^b /etc/passwd
```

Procurar por linhas que terminam com a palavra “false”:

```
1 # egrep false$ /etc/passwd
```

6.1.4 Comando fgrep

Ao utilizar o fgrep toda operação de expressão regular será ignorada, tornando o processo de localização muito mais rápido. Visualize o conteúdo do arquivo /etc/shadow:

```
1 # cat /etc/shadow
```

Agora visualize apenas as linhas que contenham o caracter $:

```
1 # fgrep $ /etc/shadow
```

6.1.5 Comando sed

O comando **sed** é utilizado para procurar e substituir padrões em texto, mostrando o resultado em stdout.

No **sed,** a expressão fica circunscrita entre barras(/). Por exemplo: Deletar as linhas comentadas do arquivo "/etc/fstab":

```
1 # sed -i '/^#/d' /etc/fstab
```

-e - Executa a expressão e comando a seguir. ^ início de linha **# -** string de busca

A letra d ao lado da expressão regular é um comando sed, que indica a exclusão de linhas contendo o respectivo padrão.

Para substituir uma string, utilize a opção **-s:** Substitua todos os caracteres "#"por "@"em /etc/fstab:

```
1 # sed -e s'/#/@/' /etc/fstab
```

Substitua agora os caracteres "/" por "@" :

```
1 # sed -e s'/\//@/' /etc/fstab
```

Observe que você tem que escapar o caracter "/", pois este é o separador dos campos.

Ou mais fácil, utilize outro separador de campos:

```
1 # sed -e s'|/|@|' /etc/fstab
```

Observe ainda que em nenhum dos casos foi efetuada a troca de todas as instâncias da linha, somente a primeira que foi encontrada em cada linha, para que se possa resolver este problema utilize a opção "g" de global:

```
1 # sed -e s'|/|@|g' /etc/fstab
```

Para efetuar a troca em uma linha específica, aponte o número da linha, por exemplo fazer a troca dos caracteres "/" por "@" na primeira linha:

```
1 # sed -e 1s'|/|@|g' /etc/fstab
```

6.1.6 Comandos cut e awk

O comando **"cut"** pode ser muito útil para conseguir listagens a partir de arquivos com separadores de colunas definidos.

Por exemplo, para conseguir o nome de todos os usuários da máquina, ou seja, a primeira coluna do arquivo **"/etc/passwd"** e também seu uid , cujo delimitador de colunas é o sinal ":", podemos usar o comando:

```
1  # cut -f1,3 -d: --output-delimiter=" " /etc/passwd  > /root/uid
```

Onde:

-f - coluna **1,3 -** coluna 1 e 3 **-d -** delimitador -**-output-delimiter-** - delimitador da saída do comando

O comando **"awk"** é um primo do "cut", mas possui mais recursos e opções, por ser uma linguagem de programação. Há situações nas quais o "cut" não conseguirá resolver o problema, para elas use **"awk".**

Para o mesmo exemplo acima, agora utilizando o **awk:**

```
1  # awk -F: '{print $1," ",$4}' /etc/passwd > /root/gid
```

Onde:

-F - delimitador

print - imprime o valor da coluna especificada:

$1 - coluna1

$4 - coluna4

“ “ - delimitador da saída do comando

O awk suporta mais opções que o cut, por exemplo executar novamente a busca anterior, mas desta vez trazer apenas os usuários que tenham uid inferior a 5:

```
1 # awk -F: '($3 <= 5) {print $1,$3}' /etc/passwd
```

Onde ($3 <= 5) equivale a:

$3 - coluna3 <= - menor ou igual == - igual >= - maior ou igual 5 - valor de comparação

6.1.7 Juntando dois arquivos em um: join e paste

Comando join

Vizualize os arquivos “/root/uid” e “/root/gid”:

```
1 # cat /root/uid
```

e

```
1 # cat /root/gid
```

O comando **“join”** (unir), concatena registros de dois arquivos de texto baseado em índices comuns entre os registros.

No caso dos arquivos vistos o índice em comum são os nomes dos usuários, vamos usar o join para unir os dois arquivos:

```
1 # join /root/uid /root/gid
```

Comando paste

O comando **“paste”,** junta os arquivos na saída padrão. Diferente do “join”, ele joga os dois arquivos lado-a-lado.:

```
1  # paste /root/uid /root/gid
```

Ainda com o “paste” podemos, usar o parâmetro “-d”, de delimitador:

```
1  # paste -d@ /root/uid /root/gid
```

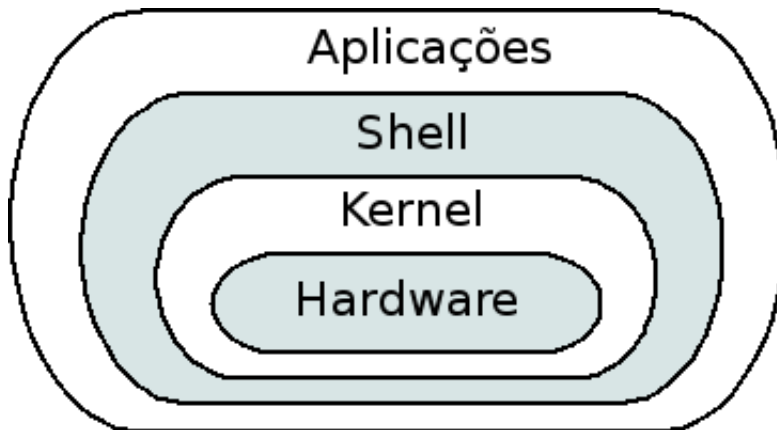
Administração da Shell

6.2 Introdução teórica

O principal meio de interação do usuário com um sistema GNU/Linux é o terminal de comandos, também conhecida como **"shell"**. Neste capítulo iremos aprender como personalizá-la e sua utilização básica.

6.2.1 O que é uma shell?

A **"shell"** é uma camada de acesso ao sistema básico, o sistema operacional do computador, que pode ser acessada tanto pelo modo gráfico, quanto em modo texto. A "shell" pode ser personalizada para atender as necessidade do usuário. Pode-se definir um idioma padrão, personalizar e automatizar processos. Nos tópicos a seguir, veremos como fazer essa personalização.

A figura abaixo ilustra como podemos posicionar a "shell" dentro do sistema.

6.2.2 Variáveis em Shell

As variáveis da "shell" têm o mesmo objetivo das variáveis que conhecemos na área da computação, ou seja, são áreas de memória que podem conter dados que serão utilizados por diversos programas. Quando estamos falando de variáveis em "shell" não precisamos nos preocupar em declará-las nem mesmo definir o seu tipo. Em "shell", uma variável é definida simplesmente atribuindo-se um valor a ela. Vejamos um exemplo: Se definirmos que "ANSWER=42", estaremos armazenando o valor "42" em um determinado endereço de memória que podemos acessar utilizando o nome que atribuímos a ele, ou seja, "ANSWER".

```
1  # ANSWER=42
```

Esse tipo de variável que acabamos de definir é conhecida como escalar e pode receber valores numéricos ou caracteres.

Para acessarmos o endereço de memória atribuído à variável "ANSWER", em "shell", devemos utilizar o operador "$" (cifrão) antes do nome da variável, ou seja, se desejarmos mostrar na tela o valor da variável "ANSWER" devemos imprimir o conteúdo armazenado no endereço de memória "$ANSWER":

```
1 # echo $ANSWER
```

O comando **“echo”** é usado para imprimir algo na tela ou direcionar para um arquivo. Isso é bastante útil para automação. Na linha de comando o “echo” é útil para inspecionar variáveis de ambiente, que são parâmetros guardados em memória e que definem o ambiente em uso.

Para imprimir algo na tela:

```
1 # echo algo
```

Vamos definir a variável “comando” com o valor igual a “ls”:

```
1 # comando=ls
```

Para verificarmos o valor da variável podemos digitar:

```
1 # echo $comando
```

Ou

```
1 # echo "$comando"
```

Para escrevermos na tela: “$comando”, digite:

```
1 # echo '$comando'
```

Para executarmos o valor da variável “comando”, digite:

```
1 # echo ‘$comando‘
```

Ou

```
1 # echo $($comando)
```

6.2.3 Variáveis Locais e de Ambiente (globais)

Quando falamos em variáveis em “shell” temos que ter em mente a divisão entre variáveis locais e de ambiente (ou globais). A diferença entre elas é que uma variável local tem visibilidade restrita, apenas ao escopo ao qual ela foi definida, e uma variável de ambiente tem visibilidade não só no escopo em que foi definida mas também em ambientes derivados.

A única diferença técnica entre variáveis locais e de ambiente é a forma de sua definição. Para definir uma variável local, basta atribuir um valor a um nome de variável. Para definir uma variável de ambiente o procedimento adiciona o comando “export”antes da definição. Abaixo mostramos exemplos de definição de variável local e de ambiente:

```
1 # LOCAL="sem export na frente"
2 # export GLOBAL="com export na frente"
```

Uma vez definidas as variáveis, podemos visualizá-las utilizando os comandos “set” e **“env”** ou **“printenv”** para variáveis locais e de ambiente, respectivamente. Com isso, se tivéssemos definido as variáveis “LOCAL” e “GLOBAL” e executássemos o comando **“set”**, veríamos as definições de ambas. Mas, se executássemos o comando “env”, veríamos apenas a definição da variável “GLOBAL”.

Visualizando:

```
# magica="abracadabra"
# echo $magica
# set
# clear
# env
Abra um terminal filho:
# bash
Não há nada na variável, pois ela não foi exportada:
# echo $magica
# exit
```

Exporte a variável:

```
# export magica
# set
# clear
# env
```

Abra um terminal filho:

```
# bash
```

Agora existe um valor para a variável:

```
# echo $magica
```

Para desabilitar utilize o comando: unset que apaga a variável:

```
1 # unset magica
2 # echo $magica
```

Para ficar permanente para todos e funcionar em qualquer terminal deve-se colocar em um dos arquivos:

```
1 /etc/profile
2 /etc/environment
```

Para ficar permanente para o usuário e funcionar em qualquer terminal deve-se colocar em um dos arquivos:

```
1  ~/.bashrc
2  ~/.bash_profile
3  ~/.bash_login
4  ~/.profile
```

Variáveis de ambiente (as globais) são muito s pois definem o comportamento da “shell” e de muitos outros programas”. Por exemplo, a forma com que o “prompt” é apresentado ao usuário é definido pela variável global “PS1”.

```
1 # echo $PS1
2 # PS1="C:\> "
```

Algumas variáveis importantes: EDITOR -> define o editor que abrirá um programa que chama o editor padrão. No debian, conseguimos fazer a alteração do editor através do comando update-alternatives –config editor, que veremos ainda no curso.

```
1 # echo $EDITOR
2 # export EDITOR=nano
```

```
3 # vipw
4 # export EDITOR=vim
5 # vipw
```

TMOUT -> tempo em segundos de inatividade para deslogar automaticamente:

```
1 # TMOUT=30
```

HOME -> home do usuário atual:

```
1 # echo $HOME
```

HISTSIZE -> tamanho do histórico de comandos:

```
1 # echo $HISTSIZE
2 # history
```

PATH -> define quais diretórios pesquisar e a ordem na qual eles são pesquisados para encontrar um determinado comando:

```
1 # su - aluno
2 $ echo $PATH
```

Vamos tentar executar um comando de rede:

```
1 $ ifconfig
```

Não conseguimos, isto ocorre ou porque não temos permissão para executá-lo, ou porque o caminho do comando não está na variável PATH do usuário. Para saber qual o caminho do binário ifconfig, digite:

```
1 $ whereis ifconfig
```

Não utilizamos o which que traz apenas o caminho do binário do comando porque ele não iria encontrar o comando ifconfig em nosso PATH já o whereis traz da localização original. Repare que o comando ifconfig tem o seu binário localizado em /sbin/ifconfig que não está no nosso PATH, quer dizer, não estava porque agora vamos adicioná-lo:

```
1 # PATH="$PATH:/sbin"
```

Vamos tentar executar o comando agora:

```
1 $ ifconfig
```

Isso não significa que você tem a permissão de root para executar os comandos de root. Tente derrubar sua placa de rede:

```
1 $ ifconfig eth0 down
```

Saber o conteúdo de algumas variáveis é muito importante: HISTSIZE HOME PS1 PATH EDITOR

6.2.4 Alias

Um recurso do “shell” que facilita muito a vida do usuário é a definição de “aliases”. Imagine que um usuário gosta de utilizar o comando “ls” sempre com os parâmetros “- -color -h -l”. O que seria dele se toda vez que fosse executá-lo tivesse que escrever o comando com todos os parâmetros?! Para resolver esse tipo de situação, basta criar um “alias” para o comando “ls” que defina que cada vez que o usuário digitar um simples “ls” ele será sucedido pelas opções definidas, e o que será executado será o comando “ls - -color -h -l”. Para criarmos esse “alias”, basta usarmos o comando abaixo:

```
1  # alias ls='ls --color -h -l'
```

Dessa forma fica fácil criar seu próprio comando. Por exemplo, um que limpe a tela:

```
1  # alias c='clear'
```

Tanto os “aliases” quanto as definições de variáveis podem ser efetuadas em linha de comando ou, para maior comodidade, utilizando arquivos apropriados para isso. Limpe a tela:

```
1 # c
```

Para visualizar todos os alias, digite:

```
1 # alias
```

Para desabilitar um alias, digite:

```
1 # unalias c
```

Tente limpar a tela novamente:

```
1 # c
```

6.2.5 Arquivos de Login

Quando uma “bash” é executada como uma “shell” de “login” interativo ela lê e executa o arquivo **“/etc/profile”,** se ele existir. Esse arquivo deve conter as configurações gerais que se aplicam a todos os usuários do sistema. Após ler o “/etc/profile”, ela irá procurar por um dos arquivos:

```
1 ~/.bash_profile
2 ~/.bash_login
3 ~/.profile
```

Esses arquivos estão na “home” do usuário, executando o primeiro que estiver disponível e tiver permissão de leitura. Além desses, executa também o arquivo “ /.bashrc”. Perceba que esses são arquivos ocultos, pois estão precedidos por um (.)

Quando a “bash” estiver sendo terminada (usuário fazendo logout), o arquivo “ /.bash_logout” será lido e executado, caso ele exista. Através deste arquivo, podemos automatizar procedimentos como por exemplo limpar a tela ao se “deslogar” do sistema.

Quando uma “bash” é chamada mas não é uma “shell de login”, o arquivo chamado será apenas o “ /.bashrc”.

Sendo assim, para criar “aliases” ou definir variáveis ou funções que sejam comuns a todos os usuários, devemos incluí-las no arquivo “/etc/profile”. Caso o usuário não queira utilizar o padrão do sistema, alterá-lo ou adicionar configurações pessoais, ele deve utilizar os arquivos “ /.bash_profile”, “ /.bash_login”, “ /.profile” ou “ /.bashrc” para isso.

Para colocar alias ou variáveis permanentes para seu usuário:

```
1 # vim ~/.bashrc
2  export TMOUT=300
3  alias ls='ls --color'
```

6.2.6 Arquivos /etc/issue e /etc/motd

Os arquivos **“/etc/issue”** e **“/etc/motd”** são usados para mostrar mensagens para os usuários e não interferem na parte operacional do sistema. A diferença entre os arquivos “/etc/issue” e “/etc/motd”, é que o primeiro exibe uma mensagem para o usuário antes que o mesmo faça “login” no sistema, enquanto o segundo exibe uma mensagem após o usuário se “logar” no sistema. Há ainda o arquivo **“/etc/issue.net”,** que contém a mensagem exibida em “logins” remotos.

Veja um exemplo de “/etc/motd” do Debian:

```
1  # cat /etc/motd
2
3 Linux aula 2.6.32-5-486 #1 Wed May 9 22:23:40 UTC 2011 i686
4
5 The programs included with the Debian GNU/Linux system are free
     software;
6 the exact distribution terms for each program are described in the
7 individual files in /usr/share/doc/*/copyright.
8
```

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
```

Veja um exemplo de “/etc/issue” no Debian:

```
 # cat /etc/issue
Debian GNU/Linux 6.0 \n \l
```

Os caracteres "(n)"e "(l)"no arquivo "/etc/issue"representam respectivamente o nome do servidor e do terminal em que o usuário está logado.

6.2.7 Tipos de shell

Para saber quais “shells” são válidos para login, basta visualizar o conteúdo do arquivo **“/etc/shells”.** A maioria das distribuições GNU/Linux traz a **“bash”** como “shell” padrão. Esta é uma evolução do **“Bourne Shell - /bin/sh”**, que tem bem poucos recursos.

Para sabermos a shell atual basta olhar a variável SHELL:

```
echo $SHELL
```

Para alterar o “shell” atual, utilizamos o comando “chsh”. Exemplo:

```
# chsh -s $(which rbash) aluno
```

A opção **$(which rbash)** é substituída pelo resultado do comando “which rbash”. Se logue como aluno e tente trocar de diretório.

```
1 # su - aluno
```

Volte ao SHELL bash:

```
1 $ chsh -s ‘which bash‘
```

A opção **‘which bash‘** é substituída pelo seu resultado “/bin/bash”. No próximo login o aluno estará no shell bash novamente.

455

Linux Essentials

www.4linux.com.br

Conteúdo

Editores de Texto

7.1 Introdução teórica

A grande maioria das configurações em sistemas GNU/Linux são feitas editando diretamente arquivos de configuração em modo texto. Para facilitar essa tarefa, é preciso conhecer alguns editores de texto, dentre eles: “vi”, “vim”, “nano”, “pico”, “mcedit”, “ed”, e “emacs” dentre outros:

- **vi -** Sem dúvida nenhuma o editor mais famoso de todos os tempos, presente em quase todas as distribuições;

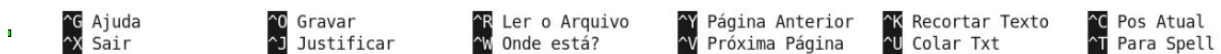
- **vim -** Uma versão melhorada do “vi”, “Vim” significa “VImproved” e traz diversas facilidades sem perder os conceitos originais do “vi”;

- **nano -** Editor padrão de muitas distribuições como Debian , CentOS esse editor é diferente do “vim” e é muito fácil de ser usado;

- **pico -** Muito parecido com o “nano”, este está presente nas distribuições Slackware e Gentoo;

- **mcedit -** Editor muito fácil e completo. Seu grande diferencial é a possibilidade da utilização do mouse, mesmo no ambiente textual;

- **ed -** O editor de textos mais simples no mundo Unix, o “ed” é um editor de linha para terminais aonde não é possível abrir uma janela de edição;

- **emacs -** Poderoso editor de "tudo", o “emacs” também é muito conhecido no mundo GNU/LINUX por fazer muitas coisas diferenciadas de um editor de texto;

Neste capítulo vamos abordar apenas a utilização dos editores “nano” e “vim”. Isso porque eles são os mais usados.

7.2 Editores de texto

7.2.1 Nano

O **“nano”** é o editor padrão de textos do Debian e Red Hat, e distribuições baseadas neles. Esse editor é muito fácil de ser usado, e sua interface é muito intuitiva e agradável. Para abrirmos o editor devemos chamar o seguinte comando:

```
1 # nano [arquivo]
```

Ao ser chamado, este editor irá apresentar um tela em branco com um rodapé semelhante a esse:

Vamos analisar essas funções:

Lembrando que "Ĝ"é igual a "Ctrl + G"e assim por diante

- **Ĝ Get Help -** Apresenta uma tela de ajuda para os mais diversos comandos e uma breve explicação sobre o editor;

- **X̂ Exit -** Sai do editor, lembrando que se o arquivo não estiver salvo, essa opção irá te pedir para salvá-lo;
- **Ô WriteOut -** Salva ou sobrescreve um arquivo;
- **Ĵ Justify -** Justifica o arquivo inteiro;
- **R̂ Read File -** Importa um arquivo;
- **Ŵ Where Is -** Procura por uma ocorrência dentro do arquivo;
- **Ŷ Prev Page -** Move o cursor para pagina anterior;
- **V̂ Next Page -** Move o cursor para próxima pagina;
- **K̂ Cut Text -** Corta a linha em que o cursor está posicionado;
- **Û UnCut Text -** Cola a linha recortada na posição atual do cursor
- **Ĉ Cur Pos -** Mostra informações sobre a posição do cursor;
- **T̂ To Spell -** Ativa a correção ortográfica. É necessário ter o comando "spell" instalado para que isso funcione;

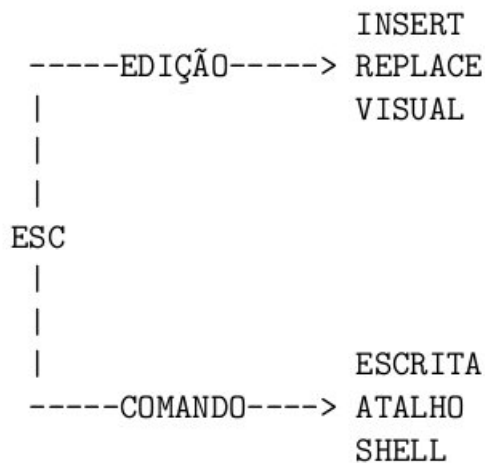
Como podemos ver, usar o editor de textos "nano", não é uma das tarefas mais difíceis no GNU/Linux. Vamos conhecer, agora, o editor "Vim".

Para definirmos qual será o editor padrão no Debian podemos usar o aplicativo "update-alternatives".

```
1 # update-alternatives --config editor
```

7.2.2 Vim

O **“Vi”** é o editor básico do GNU/Linux, e está disponível em grande parte das distribuições do GNU/Linux, mesmo naquelas que vêm em apenas um disquete. Hoje em dia, as distribuições usam uma versão mais completa e com mais recursos do que o “Vi” que é o **“Vim = VI iMproved”**. Abaixo podemos ver uma tela do editor de textos “vim”:

Ao invocar o “vim”, este entra direto para o modo “visualização”. Para modificar o arquivo, usam-se os modos de inserção, deleção e de substituição. Para voltar ao modo de visualização, sempre se usa a tecla “ESC”.

A grande maioria dos serviços em “Unix” são configurados através de arquivos de configuração, o “vim” não seria diferente. Seu arquivo de configuração fica localizado em **“/etc/vim/vimrc”**. Para configurar o seu editor de textos, basta descomentar as funcionalidades desejadas, e copiar o arquivo para o seu “home” como **“.vimrc”.**

```
1 $ cp /etc/vim/vimrc ~/.vimrc
```

```
1  # vim texto
2  # Para inserir digite: i
3  i => Entra no modo de inserção antes do cursor
4
5  # Para sair do modo de inserção digite: ESC
6
7  # Para inserir uma linha abaixo do cursor digite: o
8  o => Insere uma linha abaixo do cursor e entra no modo de inserção
9
10 # Para sair do modo de inserção digite: ESC
11
12 # Para inserir uma linha acima do cursor digite:    O
13 O => Insere uma linha acima do cursor e entra no modo de inserção
14
15 # Para sair do modo de inserção digite: ESC
16
17 # Para desfazer a última alteração digite:  u
18
19 # Para refazer digite:  CTRL+R
20
21 # Para numerar as linhas digite:  :set number
22
23 # Para copiar a segunda linha digite:  :2y
24
25 # Para colar na linha abaixo do cursor digite:   p
26
27 # Para ir para a primeira linha digite: gg
28
29 # Para colar na linha acima do cursor, "3vezes" digite: 3P
30
31 # Para salvar as alterações digite: :w
32
33 # Para sair do arquivo sem salvar digite: :q   ou   Para forçar a sa
   ída sem salvar: :q!
```

```
1 # vim texto
2 # Para ir para a última linha digite: G
3
4 # Para deletar "recortar" a linha atual digite: dd
5
6 # Para salvar e sair do arquivo: x    ou   Para forçar:   x!
```

```
1 # vim texto
2 # Delete as 5 primeiras linhas digitando:   :1,5d     ou     Com o
    cursor na primeira linha digite:   5dd   ou  d5d
3
4 # Para sair sem salvar digite:  q!
```

```
1 # vim texto
2 # buscar palavra "inser" dentro do arquivo abaixo do cursor, digite:
      /inser
3 # Para ir para a próxima ocorrência digite:   n
4 # Para ir para a ocorrência anterior digite:  N
5 # Buscar palavra "inser" dentro do arquivo acima do cursor, digite:
     ?inser
6 # Para grifar todos os resultados da busca, digite: :set hlsearch
```

Funcionalidades do Vim

Comandos básicos de inserção de texto:

i - Insere texto antes do cursor;

a - Insere texto depois do cursor;

r - Substitui texto no início da linha onde se encontra o cursor;

A - Insere texto no final da linha onde se encontra o cursor;

o - Adiciona linha abaixo da linha atual;

O - Adiciona linha acima da linha atual;

Ctrl + h - Apaga o último caractere.

Comandos básicos de movimentação:

Ctrl+f - Move o cursor para a próxima tela;

Ctrl+b - Move o cursor para a tela anterior;

H - Move o cursor para a primeira linha da tela;

M - Move o cursor para o meio da tela;

L - Move o cursor para a última linha da tela;

h - Move o cursor um caractere à esquerda;

j - Move o cursor para a próxima linha;

k - Move o cursor para linha anterior;

l - Move o cursor um caractere à direita;

w - Move o cursor para o início da próxima palavra;

W - Move o cursor para o início da próxima palavra, separadas por espaço;

b - Move o cursor para o início da palavra anterior;

B - Move o cursor para o início da palavra anterior, separadas por espaço;

0(zero) - Move o cursor para o início da linha atual;

^ - Move o cursor para o primeiro caractere não branco da linha atual;

$ - Move o cursor para o final da linha atual;

nG - Move o cursor para a linha “n”;

:n - Move o cursor para a linha “n”;

gg - Move o cursor para a primeira linha do arquivo;

G - Move o cursor para a última linha do arquivo.

Comandos básicos para localizar texto:

/**palavra -** Busca pela palavra ou caractere em todo o texto;

?palavra - Move o cursor para a ocorrência anterior da palavra;

n - Repete o último comando / ou ?;

N - Repete o último comando / ou ?, na direção reversa;

Ctrl+g - Mostra o nome do arquivo, o número da linha atual e o total de linhas.

Comandos básicos para alteração de texto:

x - Deleta o caractere que está sob o cursor;

dw - Deleta a palavra, da posição atual do cursor até o final;

dd - Deleta a linha atual, e copia o conteúdo para área de transferência;

D - Deleta a linha a partir da posição atual do cursor até o final;

:A,Bd - Deleta da linha A até a linha B, copia para área de transferência;

rx - Substitui o caractere sob o cursor pelo especificado em x;

u - Desfaz a última modificação ;

U - Desfaz todas as modificações feitas na linha atual;

J - Une a linha corrente a próxima;

yy - Copia 1 linha para a área de transferência;

yNy - Copia N linhas para a área de transferência;

p - Cola o conteúdo da área de transferência;

Np - Cola N vezes o conteúdo da área de transferência;

cc - Apaga o conteúdo da linha, e copia para área de transferência;

cNc - Apaga o conteúdo de N linhas, e copia para área de transferência;

:%s/string1/string2/g - Substitui "string1"por "string2".

Comandos para salvar o texto:

:**wq ou :x -** Salva o arquivo e sai do editor;

:w nome_do_arquivo - Salva o arquivo corrente com o nome especificado;

:w! nome_do_arquivo - O mesmo que :w, mas forçando sobrescrita;

:q - Sai do editor;

:q! - Sai do editor sem salvar as alterações realizadas.

Resumo de VI para a LPI:

:set ic => ignora case sensitive

:set number => numera as linhas

:syntax on => colore o texto

:set hlsearch => grifa o texto

:w => Salva o arquivo que está sendo editado no momento.

:q => Sai.

:wq => Salva e sai.

:x => Salva e sai.

ZZ => Salva e sai.

:w! => Salva forçado.

:q! => Sai forçado.

:wq! => Salva e sai forçado.

Deixando o vim com sua cara

No Debian o arquivo é “/etc/vim/vimrc”

 No CentOS é “/etc/vimrc”.

Adicione ao final do arquivo as opções para deixar o texto com as linhas numeradas, texto colorido e grifar as buscas encontradas:

```
1 # vim /etc/vim/vimrc
2 set number
3 syntax on
4 set hlsearch
```

455

Linux Essentials

www.4linux.com.br

Conteúdo

Gerenciamento de Pacotes em Alto Nível

8.1 Introdução teórica

8.1.1 O que é um pacote?

Os diversos programas para GNU/Linux são distribuídos em forma de pacotes específicos para cada distribuição. Neste capítulo aprenderemos um pouco sobre esses pacotes e como instalá-los e removê-los do sistema.

Pacotes são conjuntos de binários pré-compilados, bibliotecas, arquivos de controle e arquivos de configuração, que são instalados facilmente no sistema operacional. Eles podem, eventualmente, conter sistemas de listagem/checagem de dependências e "scripts" para configurações pós instalação.

Os pacotes nos sistemas baseados em Debian têm uma extensão característica: ".deb".

Já nas distribuições baseadas em RedHat, temos pacotes com a extensão característica: ".rpm".

8.1.2 Mas o que é um gerenciador de pacotes?

Um gerenciador de pacotes é um sistema para a instalação, atualização e remoção de programas em uma distribuição GNU/Linux. Parece muito simples falar em instalação de pacotes, mas temos que lembrar que é o gerenciador de pacotes quem faz toda a parte "suja" para nós.

Um pacote nem sempre depende apenas dele mesmo, ou seja, quando instalamos um programa, ele pode depender de bibliotecas de áudio, vídeo, imagens, funções e vários outros programas que precisam estar instalados antes do pacote. É esse elo entre programas que chamamos de dependências.

O trabalho feito pelo gerenciador de pacotes é interpretar a necessidade de cada um dos pacotes, para que eles possam ser instalados e/ou removidos. Para os sistemas baseados em Debian, a ferramenta a ser utilizada é o **"aptitude" ou "apt-get"**.

Já para sistemas baseados em RedHat temos uma ferramenta análoga chamada "yum".

8.2 Gerenciando Pacotes no Debian

Para gerenciarmos os pacotes no Debian, primeiramente devemos selecionar os repositórios para isso editaremos o arquivo **"/etc/apt/sources.list".**

O arquivo **"/etc/apt/sources.list"** contém os locais onde o "APT" encontrará os pacotes, a versão da distribuição que será verificada (stable, testing, unstable) e a seção que será copiada (main, non-free, contrib, non-US). Essas definições são usadas em um GNU/Linux Debian.

Segue um exemplo de arquivo de configuração:

```
# vim /etc/apt/sources.list
#mirros de segurança
deb http://security.debian.org/ squeeze/updates main contrib
deb-src http://security.debian.org/ squeeze/updates main contrib
#mirros oficiais
deb ftp://ftp.br.debian.org/debian/ squeeze main contrib non-free
deb http://linorg.usp.br/debian/ squeeze main contrib non-free
#mirror multimidia
deb http://debian-multimedia.org/ squeeze main
```

8.2.1 Instalação, Remoção e Atualização

Após fazer as configurações da lista de repositórios será necessário fazer a atualização da lista:

```
# aptitude update
```

ou

```
# apt-get update
```

Os comandos acima sincronizam a lista de pacotes disponíveis para instalação nos servidores remotos, com uma lista local. A lista local visa acelerar as consultas e pesquisas. Caso ocorra um erro de GPG, basta instalar o pacote debian-keyring.

```
# aptitude install debian-keyring
```

ou

```
1 # apt-get install debian-keyring
```

Atualize novamente o repositório:

```
1 # aptitude update
```

ou

```
1 # apt-get update
```

Vamos primeiramente utilizar a ferramenta aptitude em seu modo visual, e logo após em seu modo texto, para instalar, remover e procurar pacotes.

```
1 # aptitude
```

Para procurarmos por um pacote que desejamos instalar, podemos fazer uma busca pelo comando abaixo:

```
1  # aptitude search <argumento>
```

ou

```
1 # apt-cache search <argumento>
```

A diferença entre aptitude e apt-cache é que o “aptitude” irá trazer os resultados que contenham o argumento passado na busca no nome do programa, enquanto que o

“apt-cache” trará como resultado tanto comandos, quanto resumo do comando que contenham o argumento passado na busca.

Para buscar uma lista completa de pacotes disponíveis para Debian acesse: http://packages.debian.org

Vamos buscar por gerenciadores de janela, mas não sabemos quais existem, então prefira utilizar o apt-cache e faça a busca sempre em inglês:

```
1 # apt-cache search "display manager"
```

Se fizer a busca com o aptitude, talvez você não encontre o pacote que procura.

```
1 # aptitude search "display manager"
```

Já no caso de saber o nome do programa que você busca, você pode utilizar diretamente o aptitude, por exemplo o programa “ORCA” para deficientes visuais:

```
1 # aptitude search orca
```

O apt-cache traz mais opções, pois busca no nome do programa e no resumo do que faz o mesmo.

```
1 # apt-cache search orca
```

Agora falando de dependências de pacotes, é importante entender que os pacotes não são apenas binários mágicos, que depois de um comando de instalação estão “prontinhos” para funcionar.

A instalação de um pacote depende de vários pré-requisitos que o próprio pacote é capaz de resolver e/ou indicar ao sistema como resolver. Por exemplo, queremos instalar o pacote “kdm”, um gerenciador de displays. Após comunicarmos que queremos instalar esse pacote, o nosso gerenciador de pacotes verificará suas dependências, recomendações, conflitos e/ou apenas sugestões de tarefas, que devem ser satisfeitas.

A visualização de todas essas informações pode ser feita executando o comando:

```
1 # aptitude show <pacote>
```

ou

```
1 # apt-cache show <pacote>
```

Vamos conhecer um pouco mais sobre o “KDM” e sobre o “ORCA”:

```
1 # aptitude show kdm
```

```
1 # apt-cache show orca
```

Para a instalação de pacotes deve-se usar o comando “aptitude” ou “apt-get” com a instrução “install” e, em seguida, fornecer o nome do pacote desejado. Por exemplo, para instalar os programas “kdm, xdm, samba”, digitamos:

```
1 # apt-get install samba
```

ou

```
1 # aptitude install samba
```

As dependências são pacotes que estão diretamente ligados ao pacote que irá ser instalado, ou seja, são pré-requisitos essenciais. Se um pacote depende de outro, ambos devem ser instalados pois o programa em questão só irá funcionar se todas suas dependências estiverem supridas.

As recomendações são pacotes que não são essenciais, porém adicionam/retiram alguma função que o programa poderia ter. Por exemplo, quando instalamos o pacote "mozilla-browser" é recomendado também a instalação do pacote "mozilla-psm", que dá suporte às paginas seguras, mas este último não é obrigatório e, portanto, se não for instalado não será um impedimento na instalação do "mozilla-browser".

As sugestões são pacotes relacionados com o complemento de funcionalidade. A instalação desse pacote pode fornecer alguns complementos em relação ao pacote que está sendo instalado.

Os conflitos são pacotes que não podem ser instalados simultaneamente no sistema.

Para remover um pacote instalado deve-se usar o comando "aptitude" ou "apt-get" com a instrução "remove" e, em seguida, fornecer o nome do pacote. Por exemplo, para remover o programa "samba", digitamos:

```
1 # aptitude remove samba
```

ou

```
1 # apt-get remove samba
```

Repare que após a remoção, os arquivos de configuração do samba, ainda continuam existindo no sistema:

```
1 # ls /etc/samba
```

Instale novamente o samba:

```
1 # apt-get install samba
```

ou

```
1 # aptitude install samba
```

Agora iremos remover o samba e suas dependências, assim como seus arquivos de configuração:

```
1 # aptitude purge samba
```

Ou:

```
1 # apt-get autoremove --purge samba
```

8.2.2 Removendo pacotes que não serão mais usados

Quando você instala um pacote o apt busca das fontes listadas em **"/etc/apt/sources.list"** os arquivos necessários e os guarda em um repositório local **"/var/cache/apt/archives/"**, e então faz a instalação Em algum tempo o repositório local

pode crescer e ocupar muito espaço em disco. Felizmente o apt fornece uma ótima ferramenta para lidar com seu repositório local,o métodos "clean"do apt-get. O **"apt-get clean"** remove tudo exceto os arquivos de lock dos diretórios **"/var/cache/apt/archives/"** e **"/var/cache/apt/archives/partial/"**. Assim, se você precisar reinstalar um pacote o apt irá buscá-lo novamente.

```
1 # apt-get clean
```

8.2.3 Atualizar pacotes instalados

Para atualizar os pacotes já instalados, para a última versão que está no repositório:

```
1 # aptitude upgrade
```

ou

```
1 # apt-get upgrade
```

8.2.4 Atualização da distro

O sistema pode ser atualizado de tempos em tempos ou por questões de segurança. Para instalar todas as atualizações disponíveis, usa-se o "aptitude" com a instrução "safe-upgrade". Dependendo da velocidade de conexão, este processo pode levar bastante tempo.

```
1  # aptitude safe-upgrade
```

8.3 Gerenciamento de Pacotes em Distros baseadas em RPM

Nas distros baseadas em RedHat, o gerenciamento de pacotes é feito pelo programa “rpm”. A RedHat e Fedora disponibilizam também a ferramenta “yum”, similar em funcionalidade ao “aptitude”. Já o SUSE apresenta a ferramenta “zypper”, muito embora nesta distro recomenda-se a utilização da ferramenta “Yast” para gerenciamento de pacotes e configuração do sistema. Quando falamos de Mandriva a ferramenta da vez é o “urpmi”.

O yum (Yellow dog Update, Modified) é o gerenciador de pacotes usado por padrão no CentOS, no Fedora e no Red Hat Enterprise. O yum foi originalmente desenvolvido pela equipe do Yellow Dog (uma distribuição baseada no Red Hat, destinada a computadores com chip PowerPC) e foi sistematicamente aperfeiçoado pela equipe da Red Hat, até finalmente assumir o posto atual.

O yum trabalha de forma bem similar ao apt-get e aptitude, baixando os pacotes a partir dos repositórios especificados nos arquivos de configuração, junto com as dependências necessárias. Assim como o apt-get e aptitude, ele é capaz de solucionar conflitos automaticamente e pode ser também usado para atualizar o sistema. Essencialmente, o yum e o apt-get/aptitude solucionaram o antigo problema das dependências (um pacote precisa de outro, que por sua vez precisa de um terceiro) que atormentava os usuários de distribuições mais antigas.

Diferente do gerenciador de pacotes do Debian que toda vez que modifica a lista de repositório é necessário fazer a atualização da lista, o yum faz a atualização automaticamente cada vez que uma instalação é solicitada, checando os repositórios, baixando os headers do pacotes e calculando as dependências antes de confirmar a instalação.

Os repositórios ficam em: **/etc/yum.repos.d**, vamos adicionar o repositório do dag:

```
1 # vim /etc/yum.repos.d/dag.repo
2 [dag]
3 name=Dag RPM Repository for Red Hat Enterprise Linux
4 baseurl=http://apt.sw.be/redhat/el$releasever/en/$basearch/dag
5 gpgcheck=1
6 gpgkey=http://dag.wieers.com/rpm/packages/RPM-GPG-KEY.dag.txt
7 enabled=1
```

Procurando um programa:

```
1 # yum search <pacote>
```

Para buscar pelo software “samba”, digite:

```
1 # yum list samba
```

ou

```
1 # yum search samba
```

A diferença entre as opções “list” e “search” é que a primeira opção “list” irá trazer os resultados que contenham o argumento passado na busca no nome do programa, enquanto que a opção “search” trará como resultado tanto comandos, quanto resumo do comando que contenham o argumento passado na busca.

Obtendo informações do pacote:

Usando o **“yum”** para mostrar informações de pacotes:

```
1  # yum info <pacote>
```

Para obter informações sobre o pacote do samba:

```
1  # yum info samba
```

8.3.1 Instalando pacotes

Para instalar um pacote diretamente do repositório:

```
1  # yum install <pacote>
```

Para instalar o samba:

```
1  # yum install samba
```

8.3.2 Removendo pacotes

Para remover um pacote do sistema:

```
1  # yum remove <pacote>
```

ou:

```
1 # yum erase <pacote>
```

Remova o samba:

```
1 # yum remove samba
```

ou

```
1 # yum erase samba
```

Verifique que o diretório do samba, continua existindo no sistema:

```
1 # ls /etc/samba
```

O yum não tem uma opção “purge” como o apt-get e o aptitude, para remover as dependências e arquivos de configuração do pacote, tendo que serem removidos posteriormente.

8.3.3 Atualizando pacotes

Para atualizar os pacotes instalados, digite:

```
1 # yum update
```

455

Linux Essentials

www.4linux.com.br

Conteúdo

Instalação de Programas com DPKG e RPM

9.1 Introdução teórica

9.1.1 Pacotes Debian - DPKG

O DPKG é um programa que é a base do Sistema de Gerenciamento de Pacotes para distribuições GNU/Linux baseadas em Debian.

Criado por Ian Jackson em 1993, o DPKG é usado para instalar, remover e fornecer informações sobre os pacotes .deb.

O DPKG é uma ferramenta em linguagem de baixo nível. Front ends de alto nível são exigidos para buscar pacotes em lugares remotos ou ajudar no solucionamento de conflitos nas dependências dos pacotes.

Para esta finalidade, o Debian fornece o aptitude e o apt-get.

 Dica LPI: Não se engane !!! Na LPI é cobrado DPKG e RPM.

Estrutura de um repositório Debian:

```
pool
 \_ stable
   \_ main
     \_ a
       \_ alien
          \_ alien-<versao>.deb
       \_ a2ps
       \_ ...
     \_ b
     \_ ...
     \_ z
     \_ liba
     \_ libb
     \_ ...
     \_ libz
 \_ testing
 \_ unstable
 \_ contrib
```

9.1.2 Pacotes RPM

O RPM RedHat Package Manager - é um sistema de gerenciamento de pacotes para sistemas GNU/Linux baseados em RedHat. Ele instala, atualiza, desinstala e verifica softwares. Originalmente desenvolvido pela RedHat Linux, é agora usado por muitas distribuições como Novell - Suse que possui sua própria versão de RPM.

9.1.3 Base de dados RPM

Atrás do gerenciador de pacotes está o banco de dados RPM. Ele consiste de uma lista duplamente ligada que contém todas as informações de todos os RPM instalados. O banco de dados lista todos os arquivos que são criados ou modificados

quando um usuário instala um programa e facilita a remoção destes mesmos arquivos. Se o banco de dados fica corrompido (o que acontece facilmente se o cliente de RPM é fechado subitamente), as ligações duplas garantem que eles possam ser reconstruídos sem nenhum problema. Em computadores com o sistema operacional RedHat instalado, este banco da dados encontra-se em /var/lib/rpm.

Uma vantagem que o RPM possui sobre DPKG é que possui algumas ferramentas de verificação criptográfica com o GPG e o md5, além de verificação de integridade dos arquivos já instalados. Existe uma documentação que pode ser usada para qualquer distro baseada em RPM que pode ser encontrada em: http://www.rpm.org/RPM-HOWTO/.

9.1.4 Gerenciando Pacotes em Formato DPKG

DPKG

```
1 # dpkg --help
```

 Acesse o site a seguir e faça o download do flash player:

http://packages.debian.org/squeeze/flashplugin-nonfree

Veja as informações do pacote do flash:

```
1 # dpkg -I flashplugin-nonfree_2.8.2_i386.deb
```

Verifique se ele está instalado no sistema:

```
1 # dpkg -l <programa>
```

Ou:

```
1 # dpkg -l | grep <programa>
```

No caso:

```
1 # dpkg -l | grep flashplayer
```

Verifique quais programas estão instalados no sistema:

```
1 # dpkg -l | less
```

Para determinar qual pacote foi o responsável por instalar um binário no sistema, digite:

```
1 # dpkg -S $(which <caminho_completo_para_o_binário>)
```

EX: Qual o pacote responsável por instalar o comando ping?

```
1 # dpkg -S $(which ping)
```

Verifique o status de um pacote instalado:

```
1 # dpkg -s coreutils
```

Instale o programa flashplayer.

```
1 # dpkg -i flashplugin-nonfree_2.8.2_i386.deb
```

Verifique que o flashplayer foi instalado:

```
1 # dpkg -l flashplugin-nonfree_2.8.2_i386.deb
```

Determine onde estão instalados todos os arquivos do aplicativo flashplayer:

```
1 # dpkg -L flashplugin-nonfree | less
```

Determine onde serão instalados todos os arquivos do programa flashplayer:

```
1 # dpkg -c  flashplugin-nonfree_2.8.2_i386.deb | less
```

Remova o programa flashplayer:

```
1 # dpkg -r  flashplugin-nonfree
```

Verifique que foi removido:

```
1 # dpkg -l  flashplugin-nonfree
```

Verifique se seus respectivos arquivos também foram removidos:

```
1 # updatedb ; locate flashplugin
```

Apague seus arquivos de configuração:

```
1 # dpkg -P  flashplugin-nonfree
```

Alguns pacotes têm problemas de dependências e não são instalados, até que suas dependências sejam satisfeitas, para isso utilize o apt-get ou aptitude com a opção -f para resolver. Por exemplo, vamos tentar instalar o google-chrome, faça o download do pacote.deb:

```
1 # dpkg -i google-chrome-stable_current_i386.deb
```

O aplicativo não pode ser instalado porque existem pré-requisitos para sua instalação, para resolver estas dependências automaticamente, digite:

```
1 # aptitude -f  install
```

Ou:

```
1 # apt-get -f  install
```

9.1.5 Convertendo extensões de arquivos

Instalar o nosso conversor de pacotes:

```
1 # aptitude install alien
```

Veja se o pacote está instalado:

```
# dpkg -l alien
```

Iniciando nossos testes, precisamos de um arquivo.deb:

```
# cd /var/cache/apt/archives
# ls -lh
# aptitude clean
```

Faça o download dos pacotes necessários, para os testes:

```
# aptitude -d install sl  ccze ; ls -lh
# cp sl-<versao>.deb /opt
# cp ccze-<versao>.deb /opt
```

Entre no diretório /opt, para iniciarmos os teste com o "alien":

```
# cd /opt ; ls -lh
```

Convertendo para pacote .RPM:

```
# alien -r sl-<versao>.deb
# alien -r ccze-<versao>.deb
```

Convertendo para pacote .TGZ:

```
# alien -t sl-<versao>.deb
# alien -t ccze-<versao>.deb
```

Veja todos os arquivos criados:

```
1 # file sl*
2 # file ccze*
```

Vamos agora copiar o arquivo .rpm para a máquina Dexter para poder instalar o pacote.

```
1 #scp -P 2222 ccze*.rpm aluno@IP_SERVIDOR_DEXTER:/opt
```

9.1.6 Gerenciando Pacotes em Formato RPM

Red Hat: Em sistemas baseados em RedHat utilizamos o gerenciador de pacotes RPM.

Na máquina Dexter, verifique o que será instalado com o pacote ccze:

```
1 # cd /opt
2 # rpm -qp ccze--<versao>.rpm
```

Veja as informações do pacote, não instalado:

```
1 # rpm -qpi ccze--<versao>.rpm
```

Verifique quais arquivos serão instalados com o pacote:

```
1 # rpm -qlp ccze--<versao>.rpm
2 # rpm -ih --test --percent ccze--<versao>.rpm
```

As opções -h e –percent servem para mostrar uma barra de progressos e a porcentagem de conclusão.

Instale o programa:

```
1 # rpm -ih --percent ccze--<versao>.rpm
```

Verifique os arquivos instalados:

```
1 # rpm -qa ccze
```

Verifique quais arquivos foram instalados através do pacote:

```
1 # rpm -ql ccze
```

Verifique o que será efetuado ao removermos o pacote ccze:

```
1 # rpm -e --test ccze
```

Agora remova o ccze:

```
1 # rpm -e ccze
```

Obs.: se o pacote tiver dependências e você quiser removê-lo assim mesmo, utilize o parâmetro –nodeps.

Veja que o pacote ccze, foi removido:

```
# rpm -qa
# rpm -q ccze
```

Para realizar uma atualização de versão de algum programa podemos utilizar o comando:

```
# rpm -Uh pacote-<versao>;.rpm
```

Obs.: os parâmetros –test e –nodeps, opcionais, podem ser utilizados também.

Uma funcionalidade muito boa do RPM é a capacidade de realizar verificações de integridade dos pacotes instalados. Dessa forma, periodicamente você pode verificar se ocorreu alguma alteração no seu sistema sem você saber ou se sua máquina foi invadida, pode-se tentar identificar o que foi mexido nela.

Verifique a integridade de todos os pacotes instalados no sistema:

```
# rpm -Va
```

Vamos instalar o ccze novamente:

```
1 # rpm -ih --percent ccze--<versao>.rpm
```

Troque a permissão do binário ccze e verifique:

```
1 # chmod 777 /usr/bin/ccze
```

Execute o comando:

```
1 tail -f /var/log/syslog | ccze
```

Verifique que os log's estão saindo coloridos, o comando ccze serve para verificar log's desta maneira.

Dica LPI: O comando tail mostra por padrão as últimas 10 linhas de um arquivo, e em conjunto com a opção -f verifica em tempo real.

Verifique novamente a integridade de todos os pacotes instalados no sistema:

```
1 # rpm -Va
```

Algumas siglas da checagem:

S -> file Size differs

M -> Mode differs (includes permissions and file type)

5 -> MD5 sum differs

D -> Device major/minor number mismatch

L -> readLink(2) path mismatch

U -> User ownership differs

G -> Group ownership differs

T -> mTime differs

P -> caPabilities differ

455

Linux Essentials

www.4linux.com.br

Conteúdo

Compilando Programas

10.1 Introdução Teórica

Um dos pontos centrais do mundo GNU/Linux está baseado nas quatro liberdades básicas propostas pela FSF - Free Software Foundation, sendo elas:

- 1 liberdade de rodar o programa para qualquer propósito;
- 2 liberdade de acesso ao código fonte, estudar como ele funciona e adaptá-lo às suas necessidades;
- 3 liberdade de redistribuir cópias do software;
- 4 liberdade de melhorar o programa e distribuir essas melhorias em benefício da comunidade.

Para que essas quatro liberdades básicas sejam alcançadas é necessário que tenhamos acesso ao código fonte dos programas.

Tirando a parte ideológica, há diversas situações que exigem que recompilemos um determinado software a partir do código fonte, sendo algumas delas, quando necessitamos alterá-lo para que ele satisfaça alguma necessidade pessoal, corrigir um erro ou melhorar a segurança, o software não está disponível na forma de pacote ou simplesmente o pacote não vem compilado com alguma funcionalidade que desejamos.

10.1.1 Configure

Em geral, sempre que pegamos o código fonte de um programa ele virá com um aplicativo chamado configure que irá executar uma verificação em seu sistema a fim de verificar se ele dispõe de todos os componentes básicos para uma compilação bem sucedida.

Além disso, quando consultamos o help do configure ele irá nos mostrar todas as funcionalidades que podemos compilar com o programa e todas as funcionalidades que podemos retirar do mesmo para que ele se encaixe em nossas necessidades. Além da escolha das funcionalidades, ele nos permite informar a localização de certos componentes que por ventura não encontre.

Uma vez que o processo de configure for encerrado com sucesso, ele ira gerar um arquivo chamado Makefile, contém instruções de compilação e instalação entre outras.

10.1.2 Makefile

A Makefile em geral é criada utilizando a ferramenta configure e o objetivo desta é automatizar os processos de compilação, verificação e instalação dos softwares.

A Makefile é estruturada em seções; cada uma delas realiza alguma tarefa específica. Em geral essas Makefiles vêm com pelo menos três seções padrão: default, install e clean. Algumas podem vir com test ou check ou alguma outra que o desenvolvedor ache relevante. Por isso devemos sempre ler a documentação do programa.

A forma de utilização da Makefile é, simplesmente, utilizar o comando make com o nome de alguma das seções. Se nenhuma for especificada, ele irá executar a seção default.

Instalação:

Para que possamos instalar um software a partir de seu código fonte, o primeiro passo que temos que seguir é: fazer o download dele. Em geral fazemos isso acessando a página do desenvolvedor do programa. Neste capítulo vamos realizar a compilação do software chamado nmap, que pode ser encontrado em http://www.insecure.org. O procedimento de compilação de um programa parte do princípio que, através do código fonte do programa, qualquer um possa ter acesso ao código e gerar o binário final a partir dele. O procedimento de compilação sempre é bem parecido para todas as aplicações, porém, sempre que for compilar algum programa, devemos consultar o arquivo INSTALL ou o README que está sempre presente junto com o código fonte.

Vamos instalar os pacotes necessários:

```
1 # aptitude install make gcc g++ bzip2 gzip unzip
```

Descomprima e desempacote o código fonte do nmap no diretório apropriado entre nele:

```
1 # wget http://nmap.org/dist/nmap-5.51.tar.bz2
2 # tar xvjf nmap-5.51.tar.bz2 -C /usr/local
3 # cd /usr/local/nmap-versao
```

Qual é o primeiro passo para compilar um programa?? Ler os arquivos README e INSTALL:

```
1 # vim README
2 # vim INSTALL
```

Dica LPI:Todos esses métodos chegam ao mesmo resultado. Certfique-se de que você entende que o tar é capaz de arquivar direto para arquivos e que se pode

fazer uma versão comprimida de um arquivo tar como gzip. Para a prova você deve dominar o TAR e o GZIP, que estudamos no 450, e utilizaremos aqui.

Obs.: Nem sempre ambos os arquivos estarão presentes, mas certamente um deles sempre estará.

Agora que sabemos o que fazer, vamos executar. Para determinar quais são os parâmetros que podemos passar ao configure:

```
1 # ./configure --help
```

Como não estamos interessados na interface gráfica do nmap, podemos informar ao configure que não queremos que o nmap a utilize:

```
1 # ./configure --without-zenmap
```

Dica LPI: é muito comum, quando compilamos um programa a partir de seu código fonte, que alguns de seus requisitos não estejam presentes, ocasionando assim um erro durante a execução do configure. Quando isso ocorrer, deve-se identificar o componente que está faltando, instalá-lo e executar novamente o configure até que ele termine com sucesso. Fique atento á esse processo.

Quando o configure for executado com sucesso, podemos passar à compilação, mas antes vamos conhecer o arquivo Makefile criado pelo configure:

```
1 # vim Makefile
```

Agora sim vamos compilar o programa:

```
1 # make
```

Se não der nenhum erro de compilação, podemos prosseguir com a instalação:

```
1 # make install
```

Se tudo ocorreu bem, já é possível utilizar o novo aplicativo:

```
1 # nmap 192.168.200.254
```

Após compilarmos o programa, podemos remover os arquivos binários e de objetos que foram criados durante a compilação:

```
1 # make clean
```

Para desinstalar:

```
1 # make uninstall
```

Bibliotecas

10.2 Introdução Teórica

Hoje em dia é muito simples instalar um programa já compilado, com a ajuda de gerenciadores de pacotes como o rpm, dpkg, aptitude e outros. Mas você vai encontrar muitos programas disponíveis somente em código-fonte, e às vezes nem tão bem documentados assim. Entretanto, compilar um programa não é algo de outro mundo, não é um bicho de sete cabeças. A função destas bibliotecas lembra um pouco a dos arquivos .dll no Windows. Temos as bibliotecas estáticas e dinâmicas. As dinâmicas são usadas por vários programas e necessárias para instalar programas distribuídos em código fonte (os famosos arquivos **tar.gz, tgz e tar.bz2)**.

10.2.1 Tipos fundamentais de programas executáveis

Em sistemas Linux existem dois tipos fundamentais de programas executáveis. O primeiro é chamado de estático. Esse tipo de programa contém todas as funções que ele precisa para ser executado, em outras palavras, é completo. Devido a isso, os executáveis estáticos não dependem de nenhuma biblioteca externa para funcionar. O segundo tipo é o executável dinâmico. Mas como descobrir se um executável é dinâmico ou estático? Para isso, podemos usar o comando **ldd**, que produz uma lista de dependências.

Identificando as bibliotecas compartilhadas:

```
1 # ldd <caminho_do_executável>
```

Obs: Deve ser colocado o caminho completo do executável, não somente o nome do comando.

Para facilitar em vez de digitar o caminho completo:

```
1 # ldd $(which ls)
```

Exemplo de executável estático:

```
1 # aptitude install module-assistant
2 # ldd /usr/bin/module-assistant
3 not a dynamic executable
```

Exemplo de executável compartilhado:

```
1 # ldd /bin/ln
2 linux-gate.so.1 => (0xffffe000)
3 libc.so.6 => /lib/tls/libc.so.6 (0xb7ded000)
4 /lib/ld-linux.so.2 (0xb7f29000)
```

Verificando tamanhos:

```
1 # du -h /usr/bin/module-assistant /bin/ln
2 64K /usr/bin/module-assistant
3  40K /bin/ln
```

Note que um executável estático é bem maior que o executável dinâmico, isso ocorre pois o estático já contém o que precisa dentro do próprio executável. Obviamente, bibliotecas compartilhadas tendem a gerar executáveis menores, eles também usam menos memória, ou seja, menos espaço em disco é usado.

10.2.2 Modo Estático e Compartilhado

O modo estático é ligeiramente mais rápido, pois não precisa buscar bibliotecas em diretórios, mas consome mais espaço (dado que cada programa teria uma cópia da biblioteca dentro de si).

O modo compartilhado é ligeiramente mais lento, pois precisa sempre abrir o arquivo da biblioteca, mas ocupa menos espaço (dado que só se tem uma cópia da biblioteca) e facilita, centralizando a manutenção (se você precisar mudar a versão de uma biblioteca, não tem de recompilar o programa, basta trocar o arquivo da biblioteca). **O padrão é usar bibliotecas compartilhadas,** e geralmente é a decisão mais sábia, mas precisa que todas as bibliotecas necessárias estejam presentes no sistema para executar.

No Linux, bibliotecas estáticas têm nomes como libname.a, enquanto bibliotecas compartilhadas são chamadas libname.so.x.y.z onde x.y.z é alguma forma de número de versão.

A **última fase do desenvolvimento de um software é a biblioteca,** ou seja, reunir todas as partes fundamentais para haver execução. Existem tarefas que a maioria dos softwares iram querer realizar como abrir arquivos, por exemplo, e esse tipo de tarefa é realizada através de bibliotecas. No Linux, as bibliotecas podem ser encontradas em /lib e /usr/lib/ ou em outros diretórios.

Um exemplo real é a linguagem C, que é rica em poder de expressão, mas relativamente pobre em funcionalidades. Para construir aplicações que fazem uso de funcionalidades específicas, como interfaces gráficas, comunicação via rede, fórmulas matemáticas complexas, etc, devem ser usadas bibliotecas.

As bibliotecas mais comuns, utilizadas por todas as aplicações e utilitários do sistema, são:

libc: na verdade um grande "pacote"de bibliotecas que provê funcionalidades básicas de entrada/saída, de acesso a serviços do sistema, à rede, etc.

ld-linux: provê as funções necessárias para a carga de bibliotecas dinâmicas, durante a inicialização do programa.

Por default, **essas duas bibliotecas são automaticamente incluídas** e ligadas em todos os programas. Ao usar uma biblioteca estática, o linker encontra as partes que os módulos do programa precisam, e as cópia fisicamente no arquivo de saída executável que ele gera.

Para **bibliotecas compartilhadas**, não — em vez disso, ele deixa uma nota na saída dizendo quando este programa for executado, ele terá que carregar primeiro esta biblioteca. Diversos programas, **para não terem sempre que reinventar a roda**, usam bibliotecas, como a libc, por exemplo.

Apesar de parecer um termo complicado, trabalhar com bibliotecas compartilhadas é muito simples. Para isso é necessário saber em quais diretórios elas costumam estar, quais bibliotecas determinado binário utiliza e saber adicionar novas bibliotecas no sistema.

10.2.3 Listando Bibliotecas disponíveis

Para listar todas as bibliotecas disponíveis e a localização de cada uma, utilize o comando:

```
1 # ldconfig -p
```

Esse comando mostrará uma lista gigante das bibliotecas.

10.2.4 Localização das bibliotecas?

Por padrão, os programas instalados já adicionam as bibliotecas em seus devidos diretórios, que geralmente são: /lib, /usr/lib.

A ordem de buscas por bibliotecas no sistema é:

- O valor da variável: LD_LIBRARY_PATH

- Os diretórios especificados em /etc/ld.so.conf

- Os diretórios padrão do sistema para bibliotecas: /lib e /usr/lib

10.2.5 Adicionando novas bibliotecas ao sistema

Caso você tenha desenvolvido ou baixado alguma biblioteca nova e criou um diretório específico para guardá-las, adicione o caminho completo do diretório em que essa biblioteca se encontra no arquivo /etc/ld.so.conf e digite o comando ldconfig (sem parâmetros). Isso irá gerar o arquivo /etc/ld.so.cache, que contém informações sobre as bibliotecas compartilhadas disponíveis no sistema que ld.so ou ld-linux.so irão procurar.

Após a operação, confira o resultado com o comando **"ldconfig -p"** e veja se sua nova biblioteca consta na lista.

Outra opção é adicionar o caminho completo na variável de ambiente LD_LIBRARY_PATH, que instrui o carregador dinâmico para checar um certo diretório:

```
1 # export LD_LIBRARY_PATH=/caminho/para/bibliotecas1:/caminho/para/
    bibliotecas2
```

EX: Vamos criar um diretório para nossas bibliotecas e copiar uma biblioteca que o comando ping utiliza para lá, como este novo diretório não consta na lista de diretórios de bibliotecas, o comando não pode ser executado com sucesso, até que se adicione este novo diretório de bibliotecas:

Criando o diretório novo para guardar bibliotecas:

```
1 # mkdir /bibliotecas
```

Testando o comando ls:

```
1 # ls /etc
```

Listando as bibliotecas utilizadas pelo comando ls:

```
1  # ldd /bin/ls
2    linux-gate.so.1 =>  (0xb7816000)
3    libselinux.so.1 => /lib/libselinux.so.1 (0xb77e1000)
4    librt.so.1 => /lib/i686/cmov/librt.so.1 (0xb77d8000)
5    libacl.so.1 => /lib/libacl.so.1 (0xb77d0000)
6    libc.so.6 => /lib/i686/cmov/libc.so.6 (0xb768a000)
7    libdl.so.2 => /lib/i686/cmov/libdl.so.2 (0xb7686000)
8    /lib/ld-linux.so.2 (0xb7817000)
9    libpthread.so.0 => /lib/i686/cmov/libpthread.so.0 (0xb766d000)
10   libattr.so.1 => /lib/libattr.so.1 (0xb7668000)
```

Uma das bibliotecas compartilhadas utilizada pelo comando ls é a “/lib/librt.so.1”, repare que ele é apenas um link para a biblioteca original:

```
1 # ls -l ls -l /lib/librt.so.1
2 lrwxrwxrwx 1 root root 15 Set 16 17:14 /lib/librt.so.1 -> librt
     -2.11.2.so
```

Mova esta biblioteca para /bibliotecas

```
1 # mv  /lib/librt.so.1  /bibliotecas
```

Tente fazer o ls novamente:

```
1 ls /etc
```

ping: error while loading shared libraries: librt.so.1 cannot open shared object file: No such file or directory

Não foi possível executar o comando ls, pois a biblioteca compartilhada não pôde ser carregada por estar em um diretório que não é referenciado em /etc/ld.so.cache para os linkadores carregarem-na.

Vamos checar as bibliotecas que faltam:

```
1 # ldd /bin/ls
```

Você já imaginou ficar sem a librt.so.1? Sem essa lib, muitos recursos de movimentação pelo terminal não iram responder.

Nesses casos em que as bibliotecas estão em um diretório diferente é necessário dizer ao sistema para buscar bibliotecas lá:

Adicione o caminho completo ao arquivo /etc/ld.so.conf ou crie um arquivo em /etc/ld.so.conf.d com a extensão .conf:

```
1 # vim /etc/ld.so.conf.d/bibliotecas.conf
2
3 /bibliotecas
```

Atualize a lista de diretórios de bibliotecas com o comando:

```
1 # ldconfig
```

Verifique se seu diretório foi adicionado a lista:

```
1 # ldconfig -p | grep bibliotecas
```

Tente executar o ls:

```
1 # ls /etc
```

Agora foi possível, porque a biblioteca pôde ser encontrada, verifique:

```
1 # ldd /bin/ls
```

Dica LPI: Se apagarmos o arquivo /etc/ld.so.cache, é só executar o comando: ldconfig

455

Linux Essentials

www.4linux.com.br

Conteúdo

Introdução a Redes

11.1 Introdução teórica

Neste capítulo, iremos aprender alguns conceitos de redes que são muito importantes no nosso dia a dia em TI. Elementos como o endereço IP da máquina e a máscara de rede, são de fundamental importância quando lidamos com configuração de rede. Tão importante quanto os itens acima, é saber como funciona uma rede, aprendendo a configurar seu "gateway" e definir seu "DNS", além de descobrir técnicas que facilitam as configurações diária de rede nos nossos sistemas "UNIX".

Os protocolos "TCP/IP" antigamente eram usados como um padrão militar para troca de informações. Atualmente esses protocolos são o padrão mundial para comunicação de redes de computadores. Inclusive da Internet.

O protocolo **"TCP - Transmission Control Protocol"**, é orientado a conexões, transporta informações por meio de "handshake". Caso algum erro aconteça durante a comunicação ele, automaticamente, reenvia a informação. Esse protocolo garante o envio das mensagens. Podemos citar alguns serviços de rede que utilizam o protocolo "TCP": "SMTP", "FTP" e "Telnet". Já o protocolo "IP - Internet Protocol" descrito pela RFC 791, é responsável por estabelecer o esquema de endereçamento e pela definição de datagramas.

Apesar da nova versão da LPI ter diminuído os tópicos sobre "TCP/IP", ele ainda está presente na prova, e por isso dar atenção ao modelo OSI, é uma boa

idéia. O modelo OSI será estudado com detalhes no treinamento 451.

Acompanhe um pouco mais desse assunto no treinamento 451 da Formação 4Linux. O treinamento 452, mostra a configuração dos principais serviços.

11.1.1 Entendendo o IP

O endereçamento "IP", como dever ser chamado, é composto por 4 octetos e uma máscara, que determina quantos endereços são destinados a "host" e quantos endereços são destinados a rede.

O GNU/Linux não é diferente de outros sistemas operacionais. Para termos acesso a Internet ou a comunicação em rede também precisamos ter nosso número IP. O número IP está presente em todas as máquinas, mesmo nas que não tem conexão com a Internet.

Isso é possível porque em todo GNU/Linux há uma interface lógica, chamada "loopback" (lo) cujo endereço IP é "127.0.0.1" e que sempre deve estar devidamente configurada.

 Você pode estar se perguntando:

Mas por que raios eu poderia querer um serviço em uma máquina que não fala com o mundo externo?".

A resposta é simples: As aplicações da sua máquina utilizam este IP 127.0.0.1 para comunicação com outras aplicações internas. Além disso, você pode desenvolver suas páginas e sistemas Web e testá-las localmente. Ou mesmo testar a implantação

de um servidor de DNS ou um “proxy”, antes mesmo de colocá-lo em produção e evitar que seus usuários reclamem de algo que não funcionou direito. Em resumo, você pode fazer qualquer coisa que você queira e não necessariamente precisa ter contato com o mundo exterior.

Com essa interface configurada, todo o tipo de serviço pode ser ativado na máquina, desde um simples servidor de “ssh” até um servidor de “DNS”, passando por um servidor de páginas “Web”, afim de realizar testes antes de colocá-los em produção.

A Internet é totalmente endereçada por números IP’s, e não depende, isso mesmo, não depende em momento nenhum de um servidor de “DNS” para funcionar. O serviço de “DNS” apenas facilita o nosso acesso a Internet, permitindo que a navegação seja feita através de nomes e não de números. Isso significa que para entenderemos como a Internet funciona, precisamos entender como funcionam os números que ela utiliza os números IP.

Para configurarmos um número IP em nosso computador, precisamos também, configurar uma “Máscara” para esse número IP. A máscara de rede, também conhecida como “netmask”, é um número constituído por 32 bits, que é utilizado para separar redes, determinando quem é “host”, quem é rede e quem é “broadcast”.

- **Host -** Um endereço disponibilizado para computadores poderem acessar a rede;

- **Network -** Normalmente é o primeiro endereço da rede;

- **BroadCast -** Normalmente é o último endereço da rede, utilizado para que uma máquina possa falar com todas as outras.

11.1.2 Entendendo o gateway da rede

O “gateway” da rede é um “host” que conhece outros “hosts” que por sua vez conhece outros, e assim por diante. Complicado?

O principal papel do “gateway” é levar os pacotes “TCP/IP” para outras redes que os hosts que os originaram, não conhecem. É dessa forma que os pacotes saem de uma rede privada para um rede “Wan”. Para que os pacotes possam transitar pela Internet ou mesmo só por uma rede fechada é necessário um “gateway”. Mesmo em uma rede local, o “gateway” é a própria máquina, pois todos os “hosts” estão normalmente com a mesma configuração de IP, ou seja, mesma máscara, mesma classe de IP, etc.

11.1.3 O servidor DNS

Nesse ponto é muito importante frisar que o servidor de “DNS” não faz parte da configuração essencial de rede, pois para estarmos na Internet, basta termos um **“gateway”,** que a conheça, devidamente configurado. Lembre-se: a Internet é feita de números.

- **Usuário -** Suporte?
- **Suporte -** Sim, em que posso ajudar?
- **Usuário -** A Internet está fora do ar ... (O cara não consegue acessar o host orkut.com)

Para resolver esse probleminha basta digitar o comando:

```
1 $  ping 8.8.8.8
```

Se a resposta for positiva, você não tem um problema de link, cheque seu "DNS".

- **Suporte -** Sr. Usuário, percebi que você está acessando um site proibido pela empresa, há algo errado?
- **Usuário -** Eu?? .. Não, não, tudo bem, a Internet já está normalizada.

11.1.4 Arp e RARP

Vamos nos aprofundar um pouco mais nas teorias de redes e vamos verificar os protocolos "ARP" e "RARP". O protocolo "ARP" é utilizado para converter os endereços de rede (IP's), para os endereços físicos das interfaces - "MAC". Um exemplo clássico de usabilidade é identificar placas com o mesmo "MAC Address" na rede. Podemos conhecer todas as máquinas da rede e depois utilizar o comando "arp" para descobrir quais endereços IP tem o mesmo "MAC Address". Já o "Rarp" faz exatamente o oposto, transforma endereços físicos em endereços de rede.

11.1.5 Configurando a Rede

A configuração de rede em um sistema GNU/Linux é muito importante pois esses sistemas são, intrinsecamente, sistemas de rede. Ou seja, mesmo que não haja nenhum tipo de interface de rede, moldem ou qualquer outro dispositivo de conexão, ainda assim uma máquina GNU/Linux será um sistema de rede.

A configuração da rede baseia-se em três etapas:

- Configuração do número IP e sua máscara de rede;
- Configuração do "Gateway";
- Configuração dos servidores "DNS".

11.1.6 Configurando IP e Máscara

Além da interface “lo – loopback”, podemos configurar outras interfaces, basta que elas estejam presentes fisicamente e sejam suportadas pelo kernel. Na maior parte dos casos, a interface mais comum acaba sendo a interface **“eth0”** de “ethernet” número “0”, por ser a primeira. Para configurar e depois visualizar essas configurações em nossas interfaces de redes, utilizamos o comando **“ifconfig”.**

```
1  # ifconfig
```

Com esse comando é possível descobrir todas as interfaces presentes no sistema, mas para ter certeza que nenhuma interface está inativa adicionamos o parâmetro **“-a”.**

```
1  # ifconfig -a
```

Para atribuir um endereço IP para uma placa de rede utilizamos esta sintaxe:

```
1  # ifconfig <interface> <IP>
```

Exemplo:

```
1  # ifconfig eth0 172.16.0.100
```

Com esse comando estamos atribuindo o endereço IP 172.16.0.100 para a interface **“eth0”.**

Visualize o endereço da interface e sua máscara:

```
1 # ifconfig eth0
```

O comando “ifconfig” calcula automaticamente a máscara, mas se você precisar configurar uma máscara diferenciada, você deve usar o parâmetro “netmask”, assim:

```
1  # ifconfig eth0 172.16.0.100 netmask 255.255.0.0
```

Visualize o endereço da interface:

```
1 # ifconfig eth0
```

Caso você queira participar de uma outra rede, utilizando uma única placa de rede, crie uma interface virtual:

```
1 # ifconfig eth0:0 10.0.0.1
```

Onde: **:0** é o nome da placa de rede virtual, poderia ser também **:local, :net,** ou qualquer nome. Visualize a configuração da placa:

```
1 # ifconfig eth0:0
```

11.1.7 Configurando o gateway

Para que nossos pacotes saibam para onde ir eles precisam conhecer o IP do “gateway” da rede. O papel do “gateway” da rede é simples: ele funciona como uma saída para todos os pacotes daquela rede, para outras redes.

Para configurar o “gateway” da nossa rede utilizamos o comando **“route”** com os seguinte parâmetros:

```
1  # route add default gw IP
```

Adicionando uma rota padrão:

```
1  # route add default gw 172.16.0.1
```

Com esse comando é possível configurar a rota padrão de saída da nossa rede. Para listar todas as rotas traçadas, podemos utilizar o comando abaixo:

```
1  # route -n
```

A opção -n serve para o comando não tentar resolver os nomes, trazendo apenas os IP’s.

Com ele podemos descobrir se as rotas necessárias para que nossa rede funcione estão corretas.

Se desejarmos remover a rota padrão, devemos utilizar o comando:

```
1  # route del default
```

Esse comando se encarregará de remover a rota padrão para a saída da rede, mas lembre-se que essa rota é obrigatória no processo de configuração de rede. Tente pingar o gateway:

```
1  # ping 172.16.0.1
```

OK, configuração está correta. Agora tente pingar um site:

```
1 # ping  www.4linux.com.br
```

Não foi possível, porque? Para poder pingarmos um domínio é necessário configurarmos o DNS responsável pela resolução de nomes.

11.1.8 Configuração dos DNS Servers

Para não ter que memorizar todos os endereços IP que precisamos acessar, foi criado um serviço de rede chamado “DNS”. Este faz a tradução de nomes para endereços IP e vice-versa. Este serviço de rede será melhor detalhado no curso 452 da Formação 4Linux. Para configurar os servidores de “DNS” na máquina local, precisamos editar o arquivo de configurações de “DNS”, chamado “resolv.conf” localizado em “/etc”.

```
1  # vim /etc/resolv.conf
```

Para que a resolução de nomes funcione o conteúdo do arquivo “/etc/resolv.conf” deve ser parecido com este: nameserver 8.8.8.8

Com essa sintaxe acabamos de configurar um servidor de DNS, no caso o DNS do Google.

Tente pingar um site agora:

```
1 # ping www.4linux.com.br
```

Traceroute é uma ferramenta que permite descobrir o caminho feito pelos pacotes desde a sua origem até o seu destino.

```
1 # traceroute www.4linux.com.br
```

O comando “traceroute” pode ajudar os administradores a descobrir em que ponto da rede podemos ter um possível problema.

11.1.9 Configuração estática de rede

Tudo que vimos até agora, são configurações que podem ser atribuídas através de linha de comando (configurações dinâmicas). Porém nosso “host” deve estar devidamente configurado para que, por exemplo, após um “boot”, a máquina mantenha as configurações certas.

Para que isso aconteça temos que configurar o arquivo “/etc/network/interfaces” no Debian, assim:

```
1  # vim /etc/network/interfaces
2
3  auto lo
4  iface lo inet loopback
5
6  auto eth0
7  iface eth0 inet static
8    address 192.168.0.100
9      netmask 255.255.255.0
10   broadcast 192.168.0.255
11   network 192.168.0.0
12   gateway 192.168.0.1
```

Reinicie o serviço:

```
# invoke-rc.d networking stop
# invoke-rc.d networking start
```

No CentOS:

Os arquivos de configuração das interfaces de rede no CentOS estão localizados em “/etc/sysconfig/network-scripts/ifcfg-device”. Onde device é o nome da placa de rede.

Vamos configurar a rede do servidor CentOS para que as máquinas possam se comunicar:

```
# vim /etc/sysconfig/network-scripts/ifcfg-eth0

DEVICE=eth0
BOOTPROTO=static
ONBOOT=yes
IPADDR=192.168.0.1
NETMASK=255.255.255.0
BROADCAST=192.168.0.255
NETWORK=192.168.0.0
```

Reinicie o serviço:

```
# service network restart
```

Para ativar ou desabilitar uma placa de rede podemos usar a sintaxe, tanto no Debian quanto no CentOS:

```
1 # ifconfig eth0 up
2 # ifconfig eth0 down
```

Uma boa alternativa para habilitar e desabilitar as placas de redes, seriam os comando “ifup” e “ifdown”. Para que estes comandos funcionem as respectivas placas de redes devem estar habilitadas no arquivo de configuração de rede. Debian: /etc/network/interfaces CentOS: /etc/sysconfig/network-scripts/ifcfg-eth0 (o nome da placa poderá variar)

Exemplo:

```
1 # ifup eth0
2 # ifdown eth0
```

11.1.10 Configurando hosts e hostname DEBIAN

Podemos também configurar alguns atalhos para alguns endereços de rede. Esses atalhos são configurados dentro do arquivo “/etc/hosts”.

A sintaxe dele é:

```
1 IP   FQDN    HOSTNAME   ALIAS
```

* IP - endereço IP ex: 192.168.0.100 * FQDN - Full Qualified Domain Name = nome da máquina + domínio ex: aula.teste.com.br * Hostname - nome da máquina ex: aula * Alias - apelido (este é opcional) ex: micro100, micro100.teste.com.br

Exemplo :

192.168.0.100 aula.teste.com.br aula

Isso facilita nosso trabalho, uma vez que todos estão devidamente configurados, não precisamos mais decorar números IP.

O comando “hostname” altera dinamicamente o nome da máquina e deve ser utilizado da seguinte maneira:

```
1  #  hostname aula
```

Deslogue e logue para alterar o prompt. Para testar a resolução de nomes pelo arquivo hosts:

```
1 # hostname -i
```

A saída deverá ser igual ao seu ip: 192.168.0.100

```
1 # hostname -f
```

A saída deverá ser igual ao seu FQDN: aula.teste.com.br

```
1 # hostname -d
```

A saída deverá ser igual ao seu domínio: teste.com.br

```
1 # hostname -v
```

A saída deverá ser igual ao seu hostname: aula

Para alterar o “hostname” de maneira estática, devemos editar o arquivo **“/etc/hostname”:**

```
1 # vim /etc/hostname
2 aula
```

O comando “hostname” com sua opção “-f” (FQDN) mostra qual é o “Full Qualified Domain Name” da nossa máquina, sempre que formos configurar qualquer serviço externo em nossa máquina, o “FQDN” será a chave. <hostname>.<domainname>

11.1.11 Configurando hosts e hostname CentOS:

Podemos também configurar atalhos para endereços de rede no CentOS, usamos a mesma sintaxe que o Debian:

```
1  IP    FQDN     HOSTNAME   ALIAS
```

Acrescente uma linha para o hosts:

```
1 # vim /etc/hosts
2 192.168.0.1 aula.dexter.com.br aula dexter.com.br
```

Para alterar o nome da maquina no CentOS, altere a linha HOSTNAME do arquivo "/etc/sysconfig/network", como no exemplo abaixo:

```
1 # vim /etc/sysconfig/network
2 HOSTNAME=aula
```

Teste na máquina Debian:

Pingue o servidor CentOS:

```
1 # ping 192.168.0.1
```

Agora que já estamos com a rede configurada vamos tentar acessar a Internet.

Pingue o dns do google:

```
1 # ping 8.8.8.8
```

Agora “pingue” um site a sua escolha:

```
1 # ping www.4linux.com.br
```

455

Linux Essentials

www.4linux.com.br

Conteúdo

Gerenciamento de Processos

12.1 Introdução Teórica

Quando executamos algum comando, script ou iniciamos algum programa, o kernel atribui a ele um número de processo (PID) e passa a gerenciar a quantidade de recursos que ele irá disponibilizar para essa atividade. Como haverá sempre diversos processos rodando simultaneamente na máquina o kernel tem uma lista de processos que necessitam de recursos. Como não existe atualmente um sistema realmente multitarefa, capaz de realizar diversas atividades realmente ao mesmo tempo, o kernel cria uma fila de processos e a percorre disponibilizando recursos de máquina para cada um deles por um determinado período de tempo. Quanto melhor essa distribuição for efetuada melhor será o desempenho do sistema como um todo e mais próximo de um sistema multitarefas o sistema se parecerá.

A forma como o kernel gerência os processos é bastante inteligente e podemos sempre visualizar o status do processo num determinado instante, para determinar se ele está sendo executado neste mesmo instante ou se ele está aguardando tempo de máquina para que seja executado.

Podemos visualizar todos os processos que estão rodando em nosso sistema com o programa que tira uma foto dessa estrutura, conhecido como snaPShot vulgo ps:

```
1 # ps
```

Um simples ps não nos traz muita informação, ele possui dezenas de parâmetros,

então aqui estão os mais importantes: Para podermos ver com mais detalhes os processos temos a combinação dos seguintes parametros

```
1 # ps aux
```

Onde os parâmetros são:

a - mostra todos os processos existentes não associados com um terminal; u - exibe o nome do usuário que iniciou determinado processo e a hora em que isso ocorreu; x - exibe os processos que não estão associados a terminais;

Onde os campos são:

```
1  USER         - nome do usuário dono do processo;
2
3  UID          - número de identificação do usuário dono do processo;
4
5  PID          - número de identificação do processo;
6
7  PPID         - número de identificação do processo pai;
8
9  \%CPU        - porcentagem do processamento usado;
10
11 \%MEM        - porcentagem da memória usada;
12
13 VSZ          - indica o tamanho virtual do processo;
14
15 RSS          - Resident Set Size, indica a quantidade de memória
      usada (em KB);
16
17 TTY          - indica o identificador do terminal do processo;
18
19 START        - hora em que o processo foi iniciado;
20
21 COMMAND      - nome do comando que executa aquele processo;
```

```
22
23 PRI         - valor da prioridade do processo;
24
25 NI          - valor preciso da prioridade (geralmente igual aos
      valores de PRI);
26
27 WCHAN       - mostra a função do kernel onde o processo se encontra
      em modo suspenso;
28
29 STAT        - indica o estado atual do processo, sendo representado
      por uma letra:
```

- **D** Processo morto (usually IO);
- **R** Running (na fila de processos);
- **S** Dormindo Interruptamente (aguardando um evento terminar);
- **T** Parado, por um sinal de controle;
- **Z** Zombie, terminado mas removido por seu processo pai.

Essas letras podem ser combinadas e ainda acrescidas de:

- **>** o processo está rodando com **prioridade maior que a padrão,** tendo sido definida pelo kernel;
- **<** o processo está rodando com **prioridade menor que a padrão,** tendo sido definida pelo kernel;
- **+** o processo é **um processo pai**, ou seja, possui processos filhos;
- **s** o processo é um session leader, ou seja, possui processos que dependem dele;

- **l** o processo possui **múltiplas threads;**
- **L** o processo possui páginas **travadas** na memória;
- **N** o processo foi definido com uma **prioridade** diferente da padrão, tendo sido definida pelo usuário.

Também temos alguns programas que nos mostram os processos em execução:

```
1 # top
```

Com o top podemos ver o horário atual, quanto tempo a máquina está ligada, quantos usuários estão logados, quantos processos estão em aberto, rodando, em espera e zumbi:

```
1 Top - 10:19:09 up 1:11, 2 users, load average: 0.02, 0.07, 0.06
2 Tasks: 204 total,  2 running, 202 sleeping,  0 stopped,  0 zombie
3 Cpu(s): 7.1%us, 0.9%sy, 0.2%ni, 91.7%id, 0.0%wa, 0.0%hi, 0.0%si,
    0.0%st
4 Mem:  3530556k total, 1972072k used, 1558484k free,  232288k buffers
5 Swap: 1081340k total,     0k used, 1081340k free,  990248k cache
```

Onde os campos em CPU(s) são:

us = tempo de processamento, executado pelo usuário sy = tempo de processamento, executado pelo sistema ni = tempo de processamento, prioridade alterada pelo usuário id = ociosidade wa = wait - espera de I/O hi = interrupções de harware si = mapeamento das interrupções pelo Kernel st = steal time

A primeira linha do comando top também pode ser observada com o comando uptime:

```
# uptime
16:23:00 up  2:14,  3 users,  load average: 0.45, 0.69, 0.74
```

Onde é apresentado: 16:23:00 - hora atual up 2:14 - tempo que o sistema está ligado 3 users - número de usuários logados

load average: 0.45, 0.69, 0.74 - média de carga de processamento à 1min atrás, 5 min atrás e à 15min atrás.

Outro programa que nos ajuda a visualizar os processos é o **htop** muito mais amigável:

```
# aptitude install htop
# htop
```

Observe que com o htop você pode navegar na lista de processos.

No sistema de processos temos um tipo de processo que se chama threads esses processos são partes de um outro processo. Podemos ver esse cenário em forma de árvore com o comando:

```
# pstree
```

Um modo fácil de achar o PID de um processo é:

```
# pgrep cron
```

ou

```
1 # pidof cron
```

12.2 E como fazemos para gerenciar os processos?

Apesar do kernel gerenciar os processos, nós podemos enviar sinais a esses processos requisitando que eles alterem seu comportamento. Para isso utilizamos o alguns comandos para enviar um sinal de controle a um determinado processo.

Dica LPI: Uma listagem completa dos sinais possíveis pode ser vista na seção STANDARD SIGNALS do man 7 signal. Alguns dos sinais mais utilizados podem ser vistos a seguir:

SIGHUP (1) Term Hangup detected on controlling terminal or death of controlling process;

SIGKILL (9) Term Kill signal;

SIGTERM (15) Term Termination signal;

SIGCONT (18) Continue if stopped;

SIGSTOP (19) Stop process.

Para ver mais opções:

```
1 # man signal
```

Passando esses sinais aos processos podemos realizar tarefas desde, reiniciar o processo até encerrá-lo de forma forçada. Para gerenciarmos processos temos alguns comandos:

O comando pkill

```
1 # pkill -<signal> <programa>
```

Caso não seja passado nenhum sinal é usado o sinal 15 como padrão:

```
1 # pkill cron
```

O comando killall

```
1 # killall -s <signal> <programa>
```

Caso não seja passado nenhum sinal é usado o sinal 15 como padrão:

```
1 # killall cron
```

Os programas pkill e killall gerenciam os processos pelo nome do programa, mas também podemos gerenciar os processos pelo seu PID.

O comando kill

```
# kill -<signal> <pid>
```

Vamos **“matar”** um processo e seus processos filho: Acesse a parte gráfica e abra o aplicativo Ekiga: Aplicativos - Internet - Ekiga Verifique qual o pid do ekiga:

```
# pgrep ekiga
1267
```

Agora mate o processo:

```
# kill -9 1267
```

Veja que a janela do aplicativo, é fechada após a execução do kill.

Ainda na parte gráfica, abra dois terminais e digite em um deles:

```
# ekiga
```

Verifique que o comando fica no prompt do terminal, impossibilitando de utilizá-lo, até que eu cancele com um ctrl+c ou feche a janela do aplicativo. Vamos pausar a aplicação com o sinal de stop:

```
# kill -19 $(pgrep ekiga)
```

Tente avançar na configuração do aplicativo Ekiga, tente fechar a janela no “x”, repare que não é possível isto porque o processo está pausado. Para continuar o processo envie o sinal de continue:

```
1 # kill -18 $(pgrep ekiga)
```

Tente mexer na aplicação sem fechá-la, agora você consegue, mas repare que o terminal onde estava rodando ficou livre e o processo continua em execução, mas onde foi parar este processo?

Quando enviamos o sinal pro processo continuar, ele continuou, mas em background no terminal de onde foi executado, para visualizar este processo em background utilizamos o comando jobs.

Execute o comando no terminal que você executou o ekiga pela primeira vez:

```
1 # jobs
2 [1]+  Executando              ekiga &
```

Onde: [1]+ - é o número do job Executando - é o statusXECUTANDO ekiga - é o nome do programa & - significa que o processo está em background.

Para enviar um sinal para um processo em background utilize o kill:

```
1 # kill -<signal> %<njob>
```

Vamos terminar o processo:

```
1 # kill %1
```

Verifique que o processo foi terminado, lembrando que quando não passamos o sinal é utilizado o sinal 15 como padrão:

```
# jobs
```

Lembra que quando executamos o comando ekiga o terminal ficou inutilizável, pois se cancela-se o aplicativo fecharia? Nós podemos executar um comando para ser executado em background, liberando-se assim o terminal para uso:

Em vez de digitar ekiga e prender o terminal, execute-o em background:

```
# ekiga &
```

Agora você já sabe para poder executar qualquer programa em backgroung, coloque o caracter “&” no final do comando.

Verifique que o processo está em background:

```
# jobs
[1]+  Executando              ekiga &
```

Agora pause o processo:

```
# kill -19 %1
```

Se quisermos rodar novamente o programa ekiga, mas em foreground ou seja primeiro plano, isso mesmo, aquele que trava o terminal para sua execução, faça:

```
# fg <n job>
```

Ou seja:

```
# fg 1
```

Verifique que o terminal ficou travado para a execução do ekiga, se você cancelar o processo, o ekiga será fechado. Quando estamos executando um programa no terminal, nós podemos pausá-lo sem utilizar o kill, mas como isso? Simples basta digitar no terminal: **ctrl+z.**

```
crtl+z
```

Verifique que o processo foi pausado:

```
# jobs
[1]+  Parado                  ekiga
```

Se você quiser continuar executando-o só que em background, utilize o comando bg:

```
# bg <n job>
```

Ou seja:

```
# bg 1
```

Verifique que o ekiga está rodando em background:

```
# jobs
[1]+  Executando              ekiga &
```

Imagine agora que você vai rodar um comando que irá demorar muito tempo e você não quer deixar o terminal logado para evitar que alguém acesse o sistema, se você se deslogar o comando irá parar e não irá terminar, para resolver isso existe o comando **nohup:**

O comando nohup

O nohup ignora os sinais de interrupção de conexão durante a execução do comando especificado. Assim, é possível o comando continuar a executar mesmo depois que o usuário se desconectar do sistema.

Se a saída padrão é uma tty, esta saída e o erro padrão são redirecionados para o arquivo "nohup.out" (primeira opção) ou para o arquivo "$HOME/nohup.out" (segunda opção). Caso nenhum destes dois arquivos possam ser criados (ou alterados se já existem), o comando não é executado.

O nohup não coloca o comando que ele executa em background. Isto deve ser feito explicitamente pelo usuário.

Vamos executar um ping com o nohup e ver que ele continua sua execução mesmo após fecharmos o terminal:

```
1 # nohup ping -c 1000 4linux.com.br &
```

Enviamos 1000 ping's ao site 4linux.com.br.

Feche o terminal e após alguns (10)segundos abra novamente.

Verifique no home do usuário que você executou o comando nohup se tem o arquivo: "nohup.out".

```
1 # cat nohup.out
```

Execute novamente e verifique que o ping continua, pois o comando ainda não terminou de ser executado (1000 pings ao site 4linux.com.br):

```
1 # cat nohup.out
```

O Comando lsof

O comando lsof é um dos mais importantes comandos para quem administra sistemas Linux, principalmente na área de segurança. Este comando lista todos os arquivos abertos por todos os processos. Aqui, quando eu falo arquivo, não são apenas arquivos comuns, mas sim recursos que funcionam como arquivos (podem ser abertos, mapeados na memória, entre outros). Isso inclui bibliotecas, sockets, arquivos comuns, diretórios e por aí vai.

Em outras palavras, este comando nos fornece um mapeamento completo do que o programa está usando no sistema. Lembre-se que usando apenas o comando lsof, esta lista fica muito grande, pois mostra todos os arquivos de todos os processos. Por exemplo:

```
1 # lsof -n
2 COMMAND     PID       USER   FD      TYPE             DEVICE  SIZE/
     OFF       NODE  NAME
3 init          1       root   cwd       DIR                 8,1
     4096          2 /
4 init          1       root   rtd       DIR                 8,1
     4096          2 /
5 init          1       root   txt       REG                 8,1
     129800    6946871 /sbin/init
6 init          1       root   mem       REG                 8,1
     51712   11665519 /lib/libnss_files-2.11.1.so
```

```
7  init          1       root   mem       REG                  8,1
     43552   11665529 /lib/libnss_nis-2.11.1.so
8  init          1       root   mem       REG                  8,1
     97256   11665513 /lib/libnsl-2.11.1.so
9  init          1       root   mem       REG                  8,1
     35712   11665515 /lib/libnss_compat-2.11.1.so
10 init          1       root   mem       REG                  8,1
     1572232   11665453 /lib/libc-2.11.1.so
11 init          1       root   mem       REG                  8,1
     31744   11665567 /lib/librt-2.11.1.so
```

No exemplo acima, eu peguei apenas um fragmento do comando, indicando o que o comando bash está fazendo. Dá pra ver que bibliotecas ele está usando, onde ele está atuando, entre outros. O parâmetro "-n", que usei no exemplo acima, serve para que se o comando retornar algum endereço de rede (IP, por exemplo), ele não tente resolver com DNS, assim o retorno do comando fica mais rápido.

Alguns dos usos mais comuns incluem:

- Ver se algum processo está escutando uma porta na rede suspeita, ou conectado em algum lugar suspeito. Por exemplo, vários scripts de invasão ficam escondidos no sistema (com nomes de outros processos), conectados a servidores de IRC desconhecidos. Com o lsof, dá pra saber que estes comando estão fazendo algo que não é bem o que deveriam fazer ;) ;

- Ver que processo está usando um certo arquivo (lsof);

- Ver exatamente que tipos de conexão estão sendo feitas no sistema;

- Medir as memórias utilizadas pelos processos.

Quando lsof é chamado sem parâmetros, ele vai mostrar todos os arquivos abertos por todos os processos.

```
1 # lsof
```

Abra um arquivo com o comando vim, utilizando o usuário aluno:

```
1 $ vim arquivo
```

Agora abra outro terminal e descubra quem está utilizando o comando vim no sistema:

```
1 # lsof ‘which vim‘
```

Para visualizar apenas o número do processo:

```
1 # lsof -t ‘which vim‘
```

Nos mostrar quais os arquivos são abertos por processos cujos nomes começam pela letra "k"(klogd, kswapd ...);

```
1 # lsof -c k
```

Nos mostrar quais arquivos são abertos por processos cujo nome começa com “bash”:

```
1 # lsof -c bash
```

Nos mostrar quais os arquivos são abertos por processos cujos nomes começam por "bash", mas exclui aqueles cujo proprietário é o usuário "aluno":

```
1 # lsof -c bash -u ^ aluno
```

Nos mostrar os processos abertos pelo usuário aluno:

```
1 # lsof -u aluno
```

Nos mostrar quais arquivos estão usando o processo cujo PID é 1:

```
1 # lsof +p 1
```

Busca por todas as instâncias abertas do diretório /tmp :

```
1 # lsof +D /tmp
```

Instale o ssh e conecte-se na máquina do amigo:

```
1 #  apt-get install ssh -y
2 # ssh 192.168.200.X
```

Onde X é o ip do amigo.

Agora verifique as conexões da porta 22, que é a porta do ssh:

```
1 # lsof -i  :22
```

12.3 Definido prioridades dos processos

Como nós já sabemos o linux faz uma lista de processos e executa um a um de tempos em tempos, esta lista possui uma prioridade, o grau de importância para execução do processo.

Ela é definida conforme o programa, por exemplo um programa que depende de I/O tem maior prioridade, quem faz esse cálculo é o próprio sistema, quanto menor a prioridade de execução, maior será o uso de processamento com este processo para executá-lo mais rápido. Nós também podemos interargir nesse número de prioridade que é de -20(o mais importante) até o +19 (o menos importante), por padrão todos os processos executados como “root” tem a prioridade 0, já os processos executados como usuários comum tem a prioridade 10, o usuário root pode mudar entre -20 e +19, enquanto que o usuário comum pode mudar esta prioridade entre 0 e +19.

Para definir uma outra prioridade a um processo na hora que este irá ser executado devemos executar o comando nice, já se ele já estiver em execução utilize o renice:

Primeiro vamos executar um comando e ver seu nível de prioridade:

```
1 # ekiga
```

Para visualizar a prioridade do processo:

```
1 # ps -eo args,nice,pid | grep ekiga
2  ekiga         0 12605
3  grep ekiga        0 14116
```

-e - mostra todos os processos -o - mostra campos específicos: args - comando nice - prioridade pid - identificação do processo

Para alterar a prioridade do processo em execução, utilize o comando renice:

```
# renice <prioridade>  <PID>
```

Altere para -20 a prioridade do processo do ekiga:

```
# renice -20 12605
```

Para visualizar a mudança:

```
# ps -eo args,nice,pid | grep ekiga
 ekiga         -20 12605
 grep ekiga        0   14440
```

Termine o processo do ekiga:

```
# killall ekiga
```

Para executar um processo com a prioridade específica diferente da padrão utilize o comando nice:

```
# nice -n <NICE> <comando>
```

Execute o ekiga com prioridade inicial de +19:

```
# nice -n 19 ekiga &
```

455

Linux Essentials

www.4linux.com.br

Conteúdo

Manipulando Hardware e Dispositivos

13.1 Introdução teórica

O núcleo do sistema operacional GNU/Linux, o "kernel", se comunica com os dispositivos de uma maneira muito interessante: praticamente todos os dispositivos em GNU/Linux são representados por um arquivo correspondente dentro do sistema de arquivos. Exceção a esta regra são as placas de rede.

O local onde são armazenadas estas representações é o diretório "/dev". Uma listagem deste diretório mostrará uma série de arquivos, todos eles representando uma parte do seu computador. A interação com estes arquivos, pelo sistema operacional GNU/Linux, é feito através de pedidos e respostas que são enviados e recebido por esses arquivos especiais.

13.1.1 Explorando o /dev

Uma diferença marcante entre sistemas MS-Windows e "Unix-like" é a forma de lidar com partições e dispositivos, como unidade de disquete e CD-ROM. Em sistemas MS-Windows, desde uma partição no disco rígido a um "pen drive", o acesso é efetuado utilizando a idéia de "unidades"ou "drives", como o "drive" C: ou A: ou até mesmo uma unidade de rede. Esse tipo de conceito faz com que o usuário final não precise saber o que está por trás do funcionamento desses equipamentos, simplificando sua utilização ao preço da perda do conhecimento.

Em sistemas como GNU/Linux existe o conceito de dispositivos; praticamente tudo na máquina é tratado como sendo um dispositivo e pode ser acessado pelo seu respectivo arquivo localizado no diretório **“/dev”**. Uma exceção a isso é a interface de rede que é tratada diretamente no nível do kernel, não existindo um dispositivo (no “/dev”) associado a ela.

O diretório **“/dev”** consiste de um “filesystem” (sistema de arquivos) especial e pode ser de dois tipos: **“devfs”** ou **“udev”**. O primeiro é mais antigo, tendo sido substituído pelo segundo a partir do kernel 2.6.12. Uma das diferenças entre os dois é que no “devfs” os arquivos de dispositivos são criados uma única vez, dessa forma, o diretório “/dev” contém os dispositivos para todos os hardwares suportados pelo Linux, não importando se eles existem de fato na máquina ou não. Com o “udev” os dispositivos são criados de acordo com a disponibilidade no sistema. Dessa forma, o diretório contém apenas os arquivos de dispositivo para o “hardware” presentes na máquina.

O “UDEV” não "super popula” o diretório “/dev” do nosso sistema, além de nos proporcionar um método de configuração disponível em “/etc/udev/”.

Explorando o diretório “/dev” você irá se deparar com alguns arquivos especiais, conhecidos como arquivos de dispositivos. Os tipos existentes são:

bloco - utilizados para transferência de dados para “hardware” de armazenamento de dados, como discos rígidos;

caracter - conhecido também como "unbuffered"é utilizado para comunicação com “hardware” como “mice” e terminais;

fifo - conhecido também como “named pipe” é um dispositivo utilizado para realizar a comunicação entre dois processos independentes;

socket - um dispositivo do tipo “socket” é utilizado para criar um ponto de comunicação.

/dev/fdX Aqui é o dispositivo equivalente ao drive de disquete, onde o x corresponde a qual driver. Caso você tenha apenas um drive, esse drive vai ser o /dev/fd0. Se tiver 2 drives, o primeiro será /dev/fd0 e o segundo /dev/fd1, e por aí vai.

/dev/ttyX Quando você se loga no seu Linux, você acaba de se logar nesse terminal. Ou seja, um terminal serve para você se logar e usar uma shell (interpretador de comandos). O /dev/ttyX corresponde a cada terminal, onde X vai ser substituído pelo número do terminal (são 63 ao todo). Você também pode se deparar com /dev/ttypX. Neste caso é para terminais acessados por telnet/ssh.

/dev/ttySX Portas seriais! Na versão 2.2.x do kernel, estas portas seriais correspondem ao modem, ao mouse, e outras coisas ligadas nas 'COMs'. Veja a tabela:

Dispositivo	*Descrição*
/dev/ttyS0	COM1 (Porta serial 1)
/dev/ttyS1	COM2 (Porta serial 2)
/dev/ttyS2	COM3 (Porta serial 3)
/dev/ttyS3	COM4 (Porta serial 4)

Agora se você usa um kernel velho de versão anterior a 2.2.x, ao invés de ser /dev/ttySX, vai ser /dev/cuaX. Ou seja, você terá os equivalentes como /dev/cua0, /dev/cua1, /dev/cua2 e /dev/cua3. E estes dispositivos /dev/cuaX são usados para determinar os modems.

/dev/lpX Corresponde a porta da impressora ou porta de um serviço paralelo. X é o número correspondente a porta... 0 = LPT1 por exemplo.

/dev/plipX Esse dispositivo corresponde a uma conexão de cabo paralelo. O X será o número correspondente a porta, como no exemplo anterior.

/dev/console Este é um dispositivo especial, simbolizando os consoles (terminais não-gráficos).

/dev/null Este é um dispositivo nulo, ou seja, tudo que você mandar ou se referir a ele será descartado.

Seguindo essa classificação, os dois tipos mais comuns de serem manipulados são os de bloco e de caracter; como exemplos deles temos os "devices" referentes a dispositivos "IDE" conectados à máquina ("/dev/hda1", por exemplo) e o dispositivo de acesso ao mouse ("/dev/psaux", por exemplo).

Outros dispositivos de bloco importantes são os "SCSI" utilizados não apenas por discos "SCSI" mas também por dispositivos "USB" e "SATA", uma vez que são acessados utilizando essa emulação. O nome destes dispositivos é do tipo "/dev/sd[letra][número]" e seguem a mesma lógica dos dispositivos "IDE". Dessa forma, se houver um "HD SATA" conectado à máquina e mais nenhum outro dispositivo que utilize emulação SCSI, sua localização será o "device" "/dev/sda".

Os nomes dos dispositivos e a maneira como são representados na hierarquia do diretório "/dev" podem ser bastante confusos à primeira vista. Com um pouco de prática, a nomenclatura usada fará sentido.

Um mouse "USB" é representado pelo arquivo "/dev/input/mice", que pode ser traduzido como: dispositivo ("DEV") de entrada ("INPUT") do tipo apontador ("MICE" outro termo para "rato", em inglês). Um mouse "PS/2" segue uma nomenclatura um pouco mais complicada e é representada pelo arquivo "/dev/psaux", que pode ser interpretado como dispositivo auxiliar na porta "PS".

Para alguns dispositivos como o mouse, podemos realmente ver a interação com o arquivo que representa o dispositivo. No exemplo abaixo, usamos o comando "cat" para mostrar o conteúdo do arquivo de dispositivo de mouse (mexa o mouse depois de pressionar **"ENTER"**, após os comandos abaixo):

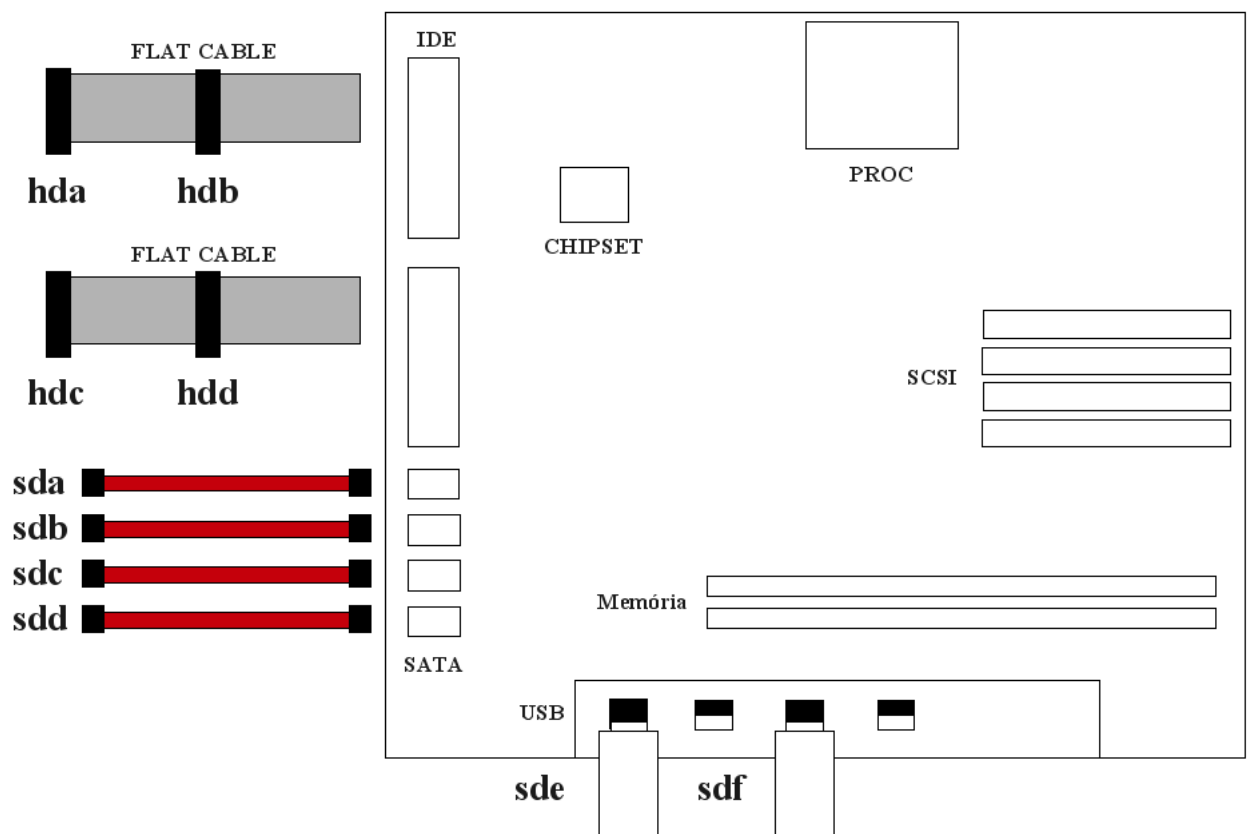
Para "mouse USB":

```
1  # cat /dev/input/mice
```

Para “mouse PS/2”:

```
1  # cat /dev/psaux
```

As saídas, ilegíveis para humanos, representam os dados que o sistema operacional GNU/Linux usa para avaliar a movimentação, posicionamento e o apertar de botões do mouse.

13.1.2 Dispositivos de armazenamento

Outro exemplo importante são os dispositivos de armazenamento principais do seu computador: os discos rígidos. Existem três tecnologias principais de discos rígidos: “IDE”, “SATA” e “SCSI”.

Outro exemplo importante são os dispositivos de armazenamento principais do seu computador: os discos rígidos. Existem três tecnologias principais de discos rígidos: “IDE”, “SATA” e “SCSI”.

Os discos “IDE” ainda são maioria no mercado, mas a tecnologia vem dando lugar ao padrão “SATA”. Tanto o padrão “IDE” como o “SATA” são considerados econômicos e mais voltados para computadores pessoais ou estações de trabalho.

Os discos do padrão “SCSI” usam uma tecnologia de acesso mais sofisticada, são geralmente mais velozes que similares “IDE” e “SATA”, além de mais robustos. São usados principalmente em servidores e máquinas de alto desempenho.

Os dispositivos “IDE” são representados na hierarquia do diretório “/dev” com um padrão que começa com “hd”. O disco rígido conectado como mestre na controladora principal será designado por “hda”. Já o escravo, nesta mesma controladora, será representado por “hdb”. Analogamente, temos “hdc” e “hdd”, respectivamente para os discos mestre e escravo conectados na controladora secundária.

Por outro lado, o padrão dos dispositivos “SATA” e “SCSI” começam por “sd”. Assim sendo, temos “sda” para o primeiro dispositivo “SATA” ou “SCSI”, “sdb” para o segundo, e assim por diante. Uma controladora “SCSI” de 8 bits pode comportar até 7 dispositivos, além da própria controladora. Para as de 16 bits, o número máximo de dispositivos é 15.

Podemos verificar o conteúdo de um disco usando novamente o comando “cat”. Para inspecionar o conteúdo do primeiro disco rígido “sata” de um computador, podemos usar o comando abaixo:

```
1 # cat /dev/sda
```

A saída gerada não parece ter nenhum sentido. Os dados mostrados são aqueles dados gravados no seu disco. Contudo, estão em uma forma que é compreensível apenas pelo sistema operacional.

Uma partição é uma divisão lógica do seu disco rígido, criada por questões de organização, conveniência, flexibilidade ou segurança. Nos sistemas baseados em representação por letras, um disco rígido sata pode ser dividido, particionado de forma a ser visto no GNU/Linux em "/dev/sda1" e "/dev/sda2".

Ou seja, a primeira partição do disco "sda" é representada por "/dev/sda1" e a segunda é representada por "/dev/sda2". Qualquer partição adicional seguirá o mesmo padrão.

13.1.3 O que é uma partição?

Uma partição é um espaço do disco que se destina a receber um sistema de arquivos ou, em um caso particular que veremos adiante, outras partições.

Tipos de partições

Existem três tipos possíveis de partições: primária, estendida e lógica.

Partições primárias

Este tipo de partição contém um sistema de arquivos. Em um disco deve haver no mínimo uma e no máximo quatro partições primárias. Se existirem quatro partições primárias, nenhuma outra partição poderá existir neste disco. As partições primárias são nomeadas da seguinte forma, caso o disco seja SATA:

- /dev/sda1

- /dev/sda2
- /dev/sda3
- /dev/sda4

Partição estendida

Isso mesmo, no singular. Só pode haver uma partição estendida em cada disco. Uma partição estendida é um tipo especial de partição primária que não pode conter um sistema de arquivos. Ao invés disso, ela contém partições lógicas. Se existir uma partição estendida, ela toma o lugar de uma das partições primárias, podendo haver apenas três.

Se houver, por exemplo, três partições no disco, sendo duas primárias e uma estendida, o esquema de nomes ficará assim:

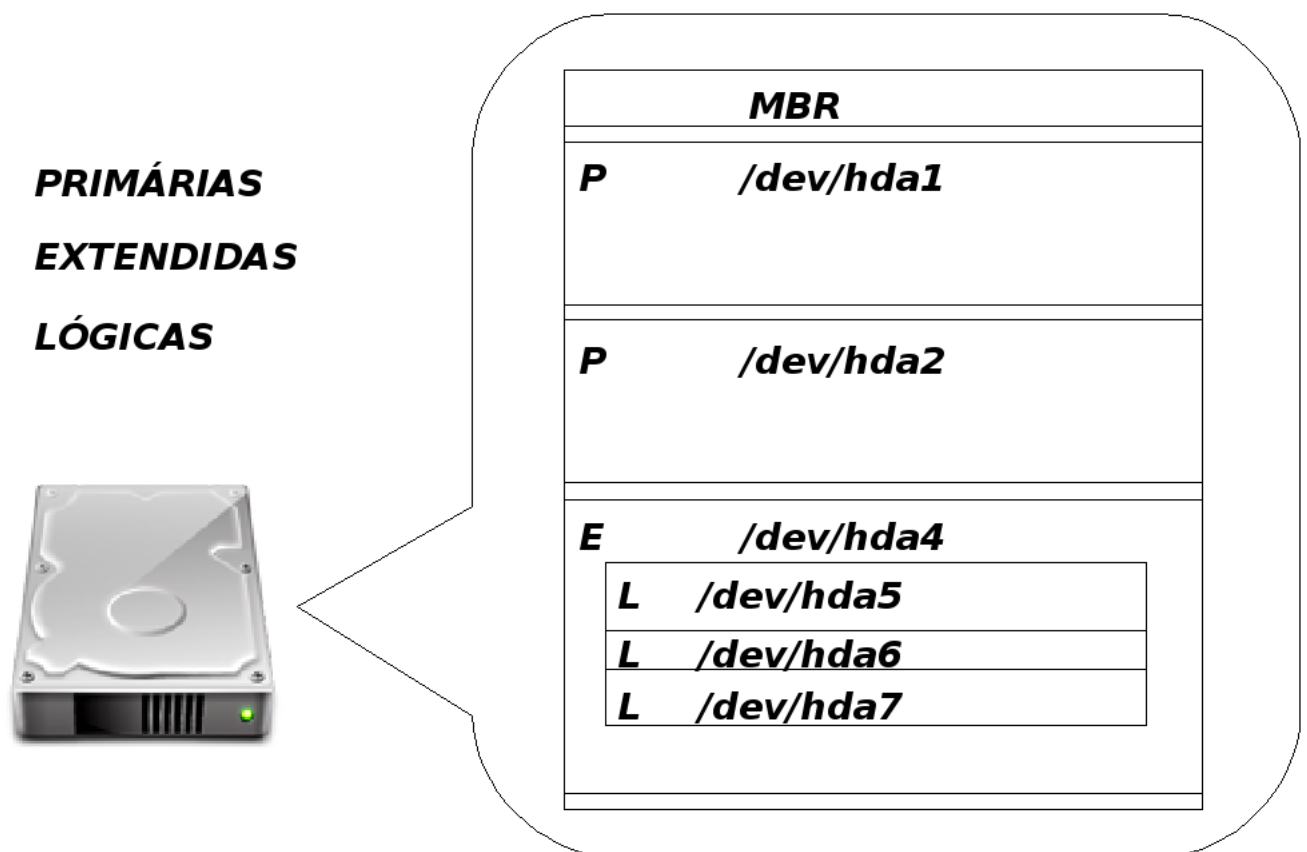
- /dev/sda1 (Primária)
- /dev/sda2 (Primária)
- /dev/sda4 (Estendida)

Partições lógicas

Também chamadas de unidades lógicas, as partições lógicas residem dentro da partição estendida. As partições lógicas são numeradas a partir de 5. Em um disco contendo duas partições primárias, a partição estendida e 3 partições lógicas, o esquema seria o seguinte:

- /dev/sda1 (Primária)
- /dev/sda2 (Primária)

- /dev/sda4 (Estendida)
- /dev/sda5 (Lógica)
- /dev/sda6 (Lógica)
- /dev/sda7 (Lógica)

Note que, neste caso, não há uma partição nomeada como /dev/sda4, pois os numeros de 1 a 4 são reservados para partições primárias e para a partição estendida.

Assim, para inspecionar o conteúdo da primeira partição, pode-se usar o comando

abaixo:

```
1 # cat /dev/sda1
```

Para interromper a saída do comando, que pode ser bastante demorada, pressione a combinação de teclas “Ctrl + c” (mantenha a tecla Ctrl pressionada e pressione a letra c). Caso a tela do seu console continue a mostrar caracteres estranhos, digite “reset”.

O último comando mostra uma saída que seres humanos não conseguem entender. Elas representam a maneira como os dados foram armazenados em “/dev/hda1”. Para que o sistema operacional GNU/Linux apresente estes dados de uma forma mais legível, é necessário solicitar ao sistema um processo de tradução. Este processo é chamado de montagem de dispositivos.

Então para que a partição “/dev/sda1” seja usada, é necessário montar esta partição em algum local e acessá-lo. Este local, que é um diretório no sistema de arquivos, é chamado de ponto de montagem. Podemos montar um dispositivo de armazenamento em qualquer diretório do sistema de arquivos, contudo, existem algumas convenções:

Dispositivos removíveis devem ser montados em /media (em outras épocas em /mnt).

Exemplos:

- **Um cdrom convencional**, representado por “/dev/cdrom” , “/dev/hdc”, “/dev/sr0” deve ser montado em “/media/cdrom”;

- **Um leitor de disquetes,** representado por “/dev/fd0”, deve ser montado em “/media/floppy”;

- **A grande maioria dos dispositivos de bloco USB**, são reconhecidos como “SCSI”, e podem ser localizados em “/dev/sda”;

- **Um Hd Sata também pode ser encontrado em "/dev/sda"**, isso pode variar, dependendo da porta "SATA" utilizada.

No caso dos discos rígidos, uma partição é montada diretamente na raiz do sistema de arquivos ou em um diretório diretamente abaixo da raiz.

13.1.4 Criando Partições no HD

Agora que já sabemos como montar um dispositivo precisamos saber como criar uma partição manualmente. Para isso, há duas ferramentas importantes, que fazem a mesma coisa, disponíveis em sistemas GNU/Linux, são elas: "fdisk" e "cfdisk".

Conhecer esses particionadores é muito importante, anote mais um: "gparted".

Adicione um novo dispositivo de armazenamento na sua máquina virtual.

Desligue a máquina, vá em configurações - armazenamento - controladora SATA - adicionar disco, crie um disco de 8GB, salve e inicie a máquina.

13.1.5 Particionamento com FDISK

O particionador "fdisk" é o mais completo dos particionadores apesar de sua interface pouco amigável.

Sintaxe:

```
1 # fdisk [dispositivo]
```

Utilizando:

```
# fdisk /dev/sdb
```

Fazendo a chamada a esse programa podemos ver a seguinte tela inicial:

```
The number of cylinders for this disk is set to 14593.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
Command (m for help):
```

Pressionando a tecla **“m”** para obtermos um “help”, veremos a seguinte saída:

```
Command (m for help): m
Command action
a toggle a bootable flag
b edit bsd disklabel
c toggle the dos compatibility flag
d delete a partition
l list known partition types
m print this menu
n add a new partition
o reate a new empty DOS partition table
p print the partition table
q quit without saving changes
s create a new empty Sun disklabel
t change a partition's system id
u change display/entry units
v verify the partition table
w write table to disk and exit
```

Para criarmos uma nova partição devemos, antes, ver se temos espaço disponível para isso, ou seja, precisamos imprimir a tabela de partições utilizando a letra “p”. Se houver espaço disponível para a criação de uma nova partição basta pressionar a letra “n” e informar o tipo da partição (primária ou estendida) e seu tamanho. **1º** crie uma partição:

```
Comando (m para ajuda): n
Comando - ação
   l   lógica (5 ou superior)
   p   partição primária (1-4)
```

2º Escolha o tipo de partição:

```
l
```

3º inicío da partição:

```
Primeiro cilindro (1-1044, padrão 1044): 1
Usando valor padrão 1
```

4º final da partição: Last cilindro, +cilindros or +sizeK,M,G (1-1044, padrão 1044): 1G

13.1.6 Particionamento com CFDISK

A ferramenta “cfdisk” não é tão completa quanto o comando “fdisk”, mas é um pouco mais amigável, ou como se diz em inglês: "user friendly". Para acessá-la basta executar o comando:

```
1  # cfdisk /dev/sdb
```

Uma vez executado esse comando, a tela do “cfdisk” se abrirá como mostrado na figura:

A utilização do “cfdisk” é bastante intuitiva, utilizando as setas para cima e para baixo você navega pela listagem das partições e, utilizando as setas para a esquerda e direita, você navega pelo menu na parte inferior da tela.

Para criar uma nova partição basta selecionar na listagem de partições a linha que contém espaço livre e entrar na opção **“New”** no menu inferior. Se ainda for possível criar partições primárias surgirá a pergunta pelo tipo da partição, caso contrário, surgirá a pergunta para especificar quanto espaço deve ser destinado para essa partição.

Crie duas partições primária de 1G cada.

Após realizar todas as alterações, escolha, no menu inferior, a opção **“Write”** para salvá-las. Uma pergunta pedindo que você confirme as alterações irá aparecer. Sua resposta deve ser “sim” ou “não” com todas as três letras! Afinal, você é o root e sabe o que está fazendo.

Criadas as partições precisamos aplicar um “filesystem” (sistema de arquivos) para que ela seja usável pelo sistema operacional.

Após salvar, visualize o conteúdo do arquivo /proc/partitions:

```
1 # cat /proc/partitions
```

Observe que a partição criada ainda não aparece, isto porque a tabela de particionamento do seu hd não foi relida, para não ter que reiniciar, instale o programa parted:

```
1 # apt-get install parted
```

Agora execute o partprobe, programa que fará a releitura do seu HD:

```
1 # partprobe
```

Verifique se sua partição foi reconhecida:

```
1 # cat /proc/partitions
```

13.1.7 Aplicando um Filesystem

Para que possamos gravar informações de forma estruturada na partição que acabamos de criar precisamos aplicar um “filesystem” a ela.

Aplicar um “filesystem”, não formatá-la!

Formatar é o processo de preparar a mídia magnética, como discos rígidos e disquetes, para receber informação. Esse tipo de preparo é de baixo nível e consiste em “desenhar” as trilhas e setores na mídia em questão. Aplicar o “filesystem” significa criar uma estrutura lógica acima dessas trilhas e setores que permita organizar seus arquivos em uma estrutura de diretórios e subdiretórios. Apesar das diferenças técnicas, os dois processos assemelham-se por apagar todo o conteúdo da partição. Portanto cuidado!

Para saber os FileSystem suportados pelo seu kernel:

```
1 # cat /proc/filesystems
```

Vamos conhecer alguns tipos de “FileSystem”:

ext2 - Um dos primeiros “FileSystem” do linux;

ext3 - Evolução do “ext2”, mas com a técnica de **“Journal”;**

ext4 - Evolução do “ext3”, mais desempenho.

reiserfs - Ótimo sistema de arquivos para arquivos menores que 4Gb;

reiser4 - Evolução “reiserfs”, mais desempenho.

xfs - Usado geralmente em banco de dados, tem suas vantagens com objetos muito grandes.

iso9660 - O sistema de arquivos padrão do CD-ROM.

msdos - Sistema de arquivos DOS.

umsdos - Sistema de arquivos para suportar arquivos DOS e Linux coexistentes.

vfat - Sistema de arquivos Windows (permite definição de nomes de arquivos com até 32 caracteres).

nfs - Sistema de arquivos remoto NFS.

proc - Sistema de arquivos Linux Process Information.

jfs (Journaling File System) : criado pela IBM para uso em servidores corporativos, teve seu código liberado.

xfs : desenvolvido originalmente pela Silicon Graphics e posteriormente disponibilizado o código fonte.

swap - Em alguns lugares ele é mencionado como um Sistema de Arquivos, mas SWAP é um espaço reservado para troca de dados com a memória RAM.

13.1.8 O que é JOURNALING?

Um sistema de arquivos com journaling dá permissão ao Sistema Operacional de manter um log (journal), de todas as mudanças no sistema de arquivos antes de escrever os dados no disco. Normalmente este log é um log circular alocado em uma área especial do sistema de arquivos.

Este tipo de sistema de arquivos tem a oferecer uma melhor probabilidade de não sofrer corrupção de dados no caso de o sistema travar ou faltar energia, e uma recuperação mais rápida, pois não necessita verificar todo o disco, somente aqueles que pertenciam a um log que não fora fechado devidamente.

Exemplos de sistemas de arquivos que suportam journaling: Ext3, Ext4, JFS, JFFS,

JFFS2, LogFS, NTFS, Reiser4, ReiserFS e XFS.

13.1.9 Aplicando um FileSystem

Para criarmos um “FileSystem” em uma partição, devemos escolher o seu tipo e utilizar o comando “mkfs” com a seguinte sintaxe:

```
1 #  mkfs -t tipo_do_FS <dispositivo>
```

 Leitura sugerida: “man mkfs”

O “FileSystem” escolhido para ser utilizado no “device” deve ser suportado pelo kernel. Para consultar quais “FileSystem” são suportados pelo kernel em uso, basta consultar o arquivo “/proc/filesystems”. Caso o “FileSystem” desejado não esteja na lista, pode-se buscar por ele nos repositórios do GNU/Linux para instalá-lo e ativá-lo como módulo do Kernel (item que estudaremos mais tarde no Treinamento 451).

Preste sempre atenção em qual partição irá aplicar o “filesystem”, pois seus seus dados serão perdidos!

Dessa forma, podemos exemplificar a criação de um “filesystem” em um dispositivo utilizando o seguinte comando:

```
1 # mkfs -t ext2 /dev/sdb1
```

13.1.10 Espaço em Disco: df

O tamanho de cada partição é definido no momento de sua criação. Mas parece bastante óbvio que depois de algum tempo será necessário determinar quanto espaço foi consumido pelos arquivos e diretórios e quanto espaço livre existe. O comando que mostra o espaço total e o espaço utilizado de todas as partições montadas chama-se **“df”**. Uma de suas características é que o cálculo dos espaços é sempre uma aproximação para a unidade de medida mais próxima, mais legível para o ser humano.

```
1 # df -h <arquivo/diretório/partição>
```

Mostra em “kilobytes” e “megabytes”:

```
1 # df -k <arquivo, diretório ou partição>
2 # df -m <arquivo, diretório ou partição>
```

13.1.11 Definindo tamanho dos objetos: du

Não é incomum precisar determinar o tamanho de um arquivo, de uma partição inteira ou de um diretório com todo o seu conteúdo. O comando “ls -alh” é capaz de informar o tamanho de arquivos, mas não de partições inteiras ou de um diretório com todo o seu conteúdo.

O comando que faz esse trabalho é o **“du”**. Veja a seguir alguns exemplos de seu uso.

```
1 # du -h <arquivo, diretório ou partição>
```

Aproxima para a unidade de medida mais próxima, mais legível para o ser humano.

Mostra em “bytes”:

```
1 # du -b <arquivo, diretório ou partição>
```

Mostra em “kilobytes”:

```
1 # du -k <arquivo, diretório ou partição>
```

Mostra em “megabytes”:

```
1 # du -m <arquivo, diretório ou partição>
```

Mostra a quantidade de “links” que arquivo/diretório/partição tem:

```
1  $ du -l <arquivo, diretório ou partição>
```

Modo resumido, ou seja, não mostra subdiretórios:

```
1 # du -s <arquivo, diretório ou partição>
```

13.1.12 Devices, UUID e Labels

Quando usamos dispositivos seguindo padrões como “/dev/hda3” ou “/dev/sda5”, estamos especificando um dispositivo que pode vir a receber outro nome. Portanto se houver alguma modificação no disco, o sistema não mais encontrará a partição especificada pois seu nome foi modificado. Uma alternativa inteligente para evitar

esse tipo de problema é utilizar o método “UUID - Universally Unique Identifier” ou o método de “Labels”.

Para descobrirmos o “UUID” de nossa partição podemos utilizar dois aplicativos: “vol_id” ou “blkid”

```
1 # vol_id -u /dev/sda1
2 f541a97e-ef19-4e47-b305-b535a75c932a
```

A opção “-u” do comando “vol_id”, nos imprime a UUID referente a uma determinada partição.

Já o comando “blkid” lista todos os dados relevantes sobre as partições do seu disco:

```
1 # blkid
2 /dev/sda1: UUID="f541a97e-ef19-4e47-b305-b535a75c932a" TYPE="ext3"
     LABEL="MAIN"
3 /dev/sda3: UUID="7C444A56444A12F6" TYPE="ntfs" LABEL="WIN"
4 /dev/sda5: TYPE="swap"
5 /dev/sda6: UUID="69ff8ed5-c09b-49b6-b21d-328e90243efa" TYPE="ext3"
     LABEL="HOME"
6 /dev/sda7: UUID="2c070d34-5c6e-4504-8d4b-9a8fa910548d" TYPE="ext3"
     LABEL="STORAGE"
7 /dev/sda8: UUID="489B-5A22" TYPE="vfat" LABEL="CENTER"
```

Há também um outro método de se descobrir essas informações. Veja o comando abaixo:

```
1 # ls -l /dev/disk/by-uuid/
2
3 lrwxrwxrwx 1 root root 10 2009-03-06 10:41 2c070d34-5c6e-4504-8d4b-9
     a8fa910548d -> ../../sda7
```

```
4 lrwxrwxrwx 1 root root 10 2009-03-06 10:41 489B-5A22 -> ../../sda8
5 lrwxrwxrwx 1 root root 10 2009-03-06 10:41 69ff8ed5-c09b-49b6-b21d
     -328e90243efa -> ../../sda6
6 lrwxrwxrwx 1 root root 10 2009-03-06 10:41 7C444A56444A12F6 ->
     ../../sda3
7 lrwxrwxrwx 1 root root 10 2009-03-06 10:41 f541a97e-ef19-4e47-b305-
     b535a75c932a -> ../../sda1
```

Mas a resposta gerada não está tão amigável quando as dos outros comandos.

Outra forma de visualizar o UUID e também o LABEL é o tune2fs para partições com arquivios de sistema ext:

```
1  # tune2fs -l /dev/sda1
2  tune2fs 1.41.11 (14-Mar-2010)
3  # aqui é mostrado o nome do label:
4  Filesystem volume name:   <none>
5  Last mounted on:          /
6  # aqui mostra o UUID da partição
7  Filesystem UUID:          7c56c1d8-7e8c-4b20-8a46-a0332355faff
8  Filesystem magic number:  0xEF53
9  Filesystem revision #:    1 (dynamic)
10 Filesystem features:      has_journal ext_attr resize_inode
      dir_index filetype needs_recovery extent flex_bg sparse_super
      large_file huge_file uninit_bg dir_nlink extra_isize
11 Filesystem flags:         signed_directory_hash
12 Default mount options:    (none)
13 Filesystem state:         clean
14 Errors behavior:          Continue
15 Filesystem OS type:       Linux
16 Inode count:              18997248
17 Block count:              75981568
18 Reserved block count:     3799078
19 Free blocks:              20316303
20 Free inodes:              18593199
21 First block:              0
```

```
22 Block size:               4096
23 Fragment size:            4096
24 Reserved GDT blocks:      1005
25 Blocks per group:         32768
26 Fragments per group:      32768
27 Inodes per group:         8192
28 Inode blocks per group:   512
29 Flex block group size:    16
30 Filesystem created:       Sun Mar 27 12:24:11 2011
31 Last mount time:          Thu Jun 30 21:11:07 2011
32 Last write time:          Sat Jun 18 18:34:53 2011
33 # total de montagem da partição:
34 Mount count:              29
35 # limite máximo de montagens antes de checar o arquivo de sistema:
36 Maximum mount count:      36
37 # data da última checagem:
38 Last checked:             Sat Jun 18 18:34:53 2011
39 # intervalo máximo para checagens:
40 Check interval:           15552000 (6 months)
41 Next check after:         Thu Dec 15 19:34:53 2011
42 Lifetime writes:          468 GB
43 Reserved blocks uid:      0 (user root)
44 Reserved blocks gid:      0 (group root)
45 First inode:              11
46 Inode size:           256
47 Required extra isize:     28
48 Desired extra isize:      28
49 Journal inode:            8
50 First orphan inode:       262883
51 Default directory hash:   half_md4
52 Directory Hash Seed:      192bfa52-e10f-4ef3-ac2f-a1acbd575b5c
53 Journal backup:           inode blocks
```

13.1.13 Rotulando uma partição:

Para colocar uma LABEL em uma partição ext3:

```
1 # tune2fs -L nome /dev/sda1
```

Cheque a alteração:

```
1 # tune2fs -l /dev/sda1
```

Partições REISERFS:

```
1 # reiserfstune -l label dispositivo
```

Partições EXT2/EXT3/EXT4:

```
1 # e2label dispositivo label
```

Partições ext4 no CentOS:

```
1 # e4label dispositivo label
```

Partições NTFS:

```
1 # ntfslabel dispositivo label
```

Partições Fat16/Fat32:

```
1 # mlabel -i dispositivo :: label
```

Partições JFS:

```
1 # jfs_tune -L label dispositivo
```

Partições XFS:

```
1 # xfs_admin -L label dispositivo
```

Partições SWAP:

```
1 # mkswap -L label dispositivo
```

13.1.14 Usando os dispositivos de armazenamento

Para termos acesso a um arquivo armazenado em mídia removível, é necessário conectar a mídia removível ao seu leitor correspondente e montar o dispositivo adequado.

O comando usado para montar dispositivos é “mount”. Sem o uso de nenhum parâmetro, ele mostra os dispositivos de armazenamento que estão montados em seu computador junto com a configuração usada para montá-los.

```
1  # mount
```

Para montar um dispositivo de armazenamento em seu ponto de montagem, o comando “mount” pode ser usado da seguinte forma:

```
1 # mount -t <tipo> -o <opções> <dispositivo> <ponto-de-montagem>
```

Para visualizar as opções de montagem:

```
1 # man mount
```

Para que seja possível acessar o conteúdo de algum dispositivo precisamos de quatro itens básicos:

- Saber o nome do dispositivo que será acessado;
- Saber o “filesystem” que ele está utilizando;
- Ter um ponto de montagem;
- Ter permissão de montagem.
- determine o filesystem:

Uma vez determinado o nome do dispositivo, podemos executar o comando **blkid** com o nome do dispositivo, e determinar qual “filesystem” ele está utilizando.

```
1 # blkid /dev/sdb1
```

Caso não obtenha resposta é porque a partição não tem um arquivo de sistema. - determine o ponto de montagem:

Se não existir um ponto de montagem, basta criar um diretório vazio no local apropriado, em geral no "/media" ou "/mnt" e executar o comando para montá-lo.

Vamos criar uma imagem ISO para simular um arquivo de cdrom:

```
1 # apt-get install genisoimage
```

- genisoimage é o programa que cria as imagens em diversos protocolos;
- -R é o protocolo para o tipo de extensão Rock Ridge, comumente usado no Linux;
- -J é o protocolo Joliet comumente usado no Windows;
- -o indica o nome do arquivo de saída;
- -l permite mais de 31 caracteres para o nome do arquivo, pode ser que o MS-DOS não consiga enxergar estes caracteres, já que ele trabalha com um protocolo 8.3;
- -V especifica uma identificação para o CD (LABEL);
- -v modo verbose;
- -pad este parâmetro é necessário em muitos OSs, inclusive no Linux, ele é acionado para evitar erros de entrada e saída;

Vamos criar um arquivo ISO, ou seja, o tipo de arquivo para CD-ROM. Para isso utilizaremos o arquivo dd pra gerar um arquivo e genisoimage para aplicar o sistema de arquivo ISO9660:

```
1 # dd if=/dev/zero of=/arquivo.iso bs=1024 count=10000
```

if = input file - origem

of=output file - destino

bs= block size - tamanho do bloco

count= count - contagem de blocos

Aplicando o sistema de arquivo ISO9660:

```
1 # genisoimage -o /arquivo.iso /arquivo
```

Montando a partição:

```
1  # mount -o loop -t iso9660 /arquivo /media/cdrom
```

A opção -o “loop” é utilizada quando se quer montar um arquivo ISO que está localizado no HD.

Para desmontar um dispositivo, o comando usado é “umount”. Neste caso é possível usar como parâmetro o ponto de montagem ou o próprio dispositivo.

Visualizando que está montado:

```
1 # mount
```

OU visualizando o arquivo /etc/mtab que lista os sistemas de arquivos montados atualmente no sistema. Sua função é idêntica ao /proc/mounts:

```
1 # cat /etc/mtab
```

Cada linha representa um sistema de arquivo correntemente montado e contém os seguintes campos (da esquerda para a direita):

- A especificação do dispositivo
- O ponto de montagem
- O tipo de sistema de arquivo

Se o sistema de arquivo está montado como somente-leitura (ro, read-only) ou leitura e gravação (rw, read-write), junto às outras opções do ponto de montagem

- Dois campos não-usados contendo zeros

OU visualizando o arquivo /proc/mounts para podermos ver o status de todos os sistemas de arquivo montados:

```
1 # cat /proc/mounts
```

OU por último o comando df visto anteriormente:

```
1 # df
```

Desmontando:

Como usuário aluno abra outro terminale acesse o diretório /media/cdrom:

```
1 $ cd /media/cdrom
```

Agora tente desmontar como root:

```
# umount /media/cdrom
```

ou de forma equivalente:

```
 # umount /arquivo.iso
```

Verifique que não foi possível, isto porque existe alguém acessando o diretório, para descobrir quais os processos que estão utilizando o /media/cdrom faça:

```
# fuser <arquivo/diretótio>
```

-i : pede confirmação antes de matar um processo (usado junto com a opção -k).

-k : mata os processos que estão acessando o arquivo/diretório especificado.

-u : identifica o usuário de cada processo.

Ou seja:

```
# fuser -u /media/cdrom
/media/cdrom 1350e(aluno)
```

A letra que aparece logo após o PID representa o tipo de acesso, onde podemos ter, por exemplo:

c : diretório atual (a partir do qual o processo foi inicializado).

e : arquivo sendo executado pelo processo.

r : diretório raiz do sistema (ponto de inicialização do processo).

Verifique que o usuário aluno está acessando o diretório e por isso não é possível desmontar o volume, após descobrir, mate o processo:

```
1 # fuser -k /media/cdrom
```

Agora tente desmontar o volume:

```
1 # umount /media/cdrom
```

455

Linux Essentials

www.4linux.com.br

Conteúdo

Gerenciamento Avançado de Partições

14.1 Introdução Teórica

O gerenciamento de partições no Linux envolve tanto desempenho com segurança, quando manipulamos partições comuns e criptografadas. Neste cenário de infra-estrutura, o Administrador tem acesso a diversas ferramentas para conversão de sistema de arquivos sem perdas de dados e configuração de mapeamentos manuas e automaticos de partições comuns e criptografadas.

14.1.1 Migrando de Filesystem ext sem perder dados:

Criar um ponto de montagem:

```
1 # mkdir /teste
```

Montar o dispositivo criado anteriormente:

```
1 # mount -t ext2 /dev/sdb1 /teste
```

Copiar dados para a partição:

```
1 # cp -r /etc/* /teste
```

Desmontar a partição:

```
1 # umount /teste
```

Vamos agora converter para ext3, sem perder os dados:

```
1 # tune2fs -j /dev/sdb1
```

Montando a partição:

```
1 #  mount /dev/sdb1 /teste
```

Visualize o tipo de filesystem com o comando mount:

```
1 # mount
```

Visualizar um arquivo para ver que não corrompeu:

```
1 # cat /teste/fstab
```

Desmontar dispositivo:

```
1 # umount /teste
```

Vamos agora converter de ext3 para ext4, sem perder os dados:

Debian:

```
1 tune2fs -O extents,uninit_bg,dir_index /dev/sdb1
```

Checar arquivo do sistema:

```
1 e2fsck -pf /dev/sdb1
```

CentOS:

```
1 tune4fs -O extents,uninit_bg,dir_index /dev/sdb1
```

Checar arquivo do sistema:

```
1 e4fsck -pf /dev/sdb1
```

Monte o sistema de arquivo:

```
1 # mount /dev/sdb1 /teste
```

Verifique o tipo de sistema de arquivos:

```
1 # mount
```

Verifique se a partição foi montada:

```
1 # mount
2  # df -h
3  # cat /etc/mtab
4  # cat /proc/mounts
```

Visualizar um arquivo para ver que não corrompeu:

```
1 # cat /teste/fstab
```

Desmontar dispositivo:

```
1 # umount /teste
```

Filesystems podem ser grandes aliados na prova, principalmente no tópico “migração de filesystems”. Lembre-se da migração mais comum de filesystems: de ext2 para ext3

14.1.2 Gerenciar partições SWAP

Este tipo de partição é utilizado para fornecer suporte a memória virtual ao GNU/Linux em adição a memória RAM instalada no sistema. Este tipo de partição é identificado pelo tipo 82 nos programas de particionamento de disco para Linux. Somente os dados na memória RAM são processados pelo processador, por ser mais rápida. Desta maneira quando você estiver executando um programa e a memória RAM começar a encher, o GNU/Linux moverá automaticamente os dados que não estão

sendo utilizados para a partição Swap e libera a memória RAM para continuar carregando os dados necessários pelo programa. Quando os dados movidos para a partição Swap são solicitados, o GNU/Linux move os dados da partição Swap para a Memória. Por este motivo a partição Swap também é chamada de Área de Troca ou memória virtual. A velocidade em que os dados são movimentados da memória RAM para a partição é muito alta.

Vamos aproveitar a partição criada anteriormente e vamos aplicar o Swap à ela:

```
 # mkswap /dev/sdb2
```

Ative essa nova partição de "swap":

```
# swapon /dev/sdb2
```

Para visualizar as partições swaps ativas:

```
# cat /proc/swaps
```

ou

```
# swapon -s
```

O comando swap tem a opção -p que habilita a prioridade: -p, –priority <nº>

Quanto maior o número maior a prioridade que pode variar entre 0 e 32767.

Para desabilitar a partição swap:

```
# swapoff /dev/sdb2
```

Para visualizar as partições swaps ativas:

```
# cat /proc/swaps
```

Ou:

```
# swapon -s
```

Dê um nome para a partição:

```
# mkswap -L SWAP /dev/sdb2
```

14.1.3 Montagem automática de Filesystem no boot

Na seção “mount” você aprendeu a montar um dispositivo de forma completa e manual, entretanto, há um arquivo que facilita a nossa vida: “/etc/fstab”. Nele devem estar as informações a respeito da montagem de todos os “filesystems” do sistema, veja um exemplo a seguir:

```
<file system> <mount point><type> <options> <dump> <pass>
proc          /proc     proc defaults 0   0
/dev/sda1  /boot  ext3  defaults  0   1
/dev/sda2  /    ext3  defaults,errors=remount-ro 0 2
/dev/sda5     /usr     ext3 defaults  0    2
/dev/sda6   /var  ext3  defaults  0          2
/dev/sda7 /tmp  ext3  defaults  0    2
```

```
8  /dev/sda8 /home ext3  defaults  0    0
9  # exemplo UUID:
10 UUID=be35a709-c787-4198-a903-d5fdc80ab2f8 /teste defaults 0 2
11 # exemplo LABEL:
12 LABEL=SWAP  none       swap sw    0    0
```

As informações que devem estar nesse arquivo, de acordo com o número da coluna, são:

Localização do "filesystem", em geral o "device" ou endereço de rede;

Ponto de montagem;

Tipo do filesystem: ext3, reiserfs, xfs, etc;

Opções de montagem: defaults = rw, suid, dev, exec, auto, nouser e async. Ver "man mount";

- Aceita os valores 0 ou 1 e informa que, havendo um sistema de backup **(dump)** configurado, deverá ser feito o seu backup;

- Aceita os valores de 0 a 2 e informa que deverá ser realizada a checagem **(pass)** de integridade do sistema de arquivos. O valor zero desativa a funcionalidade, o valor 1 deve ser especificado apenas para o "/" e o valor 2 deve ser especificado para quaisquer outros sistemas de arquivos.

Sendo assim, o **"fstab"** armazena as informações dos dispositivos comumente acessados, como as partições do sistema, discos removíveis e alguns dispositivos USB. Entretanto não mostra informação alguma a respeito de quais dispositivos estão montados neste exato momento.

Essa informação pode ser obtida no arquivo "/etc/mtab" ou no "/proc/mounts". Ambos os arquivos são uma tabela atualizada em tempo real e que mostra quais

dispositivos estão montados e com quais parâmetros.

14.1.4 Mostrar o uso de memória RAM: free

O comando “free” mostra o consumo de memória “RAM” e os detalhes sobre uso de memória virtual (SWAP):

```
# free
```

Mais detalhes:

```
# free -m
```

A saída do comando será alguma coisa parecida com:

```
 total used free shared buffers cached
Mem: 2066856 950944 1115912 038920 342612
-/+ buffers/cache: 569412 1497444
Swap: 570268 0 570268
```

Você pode olhar os arquivos do /proc também: Informações de memória:

```
# less /proc/meminfo
```

Informações de swap:

```
# less /proc/swap
```

14.1.5 Introdução a partições criptografadas

Para manipular partições criptografas, vamos utilizar o LUKS (Linux Unified Key Setup) que fornece um padrão de formato de disco para partições criptografadas, e facilita a compatibilidade entre distribuições. Este padrão permite a múltiplos usuários/senhas, a revogação efetiva de senha e fornece segurança adicional contra ataques de baixa entropia.

O que é preciso para utilizar o LUKS em minha distribuição?

- Instale em sua distribuição o pacote cryptsetup que fornece comandos para ativar e configurar dispositivos criptografados;
- Carregue os módulos dm_crypt e dm_mod utilizados pelo dispositivo mapeador de alvos dm_crypt do kernel Linux.

Na maquina Debian use o comando abaixo para instalar o pacote:

```
1 # aptitude install cryptsetup -y
```

Reinicie a maquina para carregar os módulos ou use o comando modprobe:

```
1 # modprobe dm_crypt
2 # modprobe dm_mod
```

Liste os módulos carregados através do comando lsmod:

```
1 # lsmod | grep dm
2 dm_crypt               9196  0
3 dm_mod                46122  1 dm_crypt
```

14.1.6 Criar partições criptografadas

Se vamos criptografar uma partição que já está montada e já contém dados, temos que fazer um Backup pois na construção do sistema de criptografia, os dados serão perdidos. Caso seja uma partição que ainda não contenha dados esse procedimento é mais tranquilo. Como exemplo utilize alguma partição sem nenhum sistema de arquivo aplicado.

Antes de podermos abrir uma partição encriptada precisamos inicializá-la. Este procedimento sera feito através do comando cryptsetup, que gerencia partições criptografas, permitindo operações de criação, remoção, redimensionamento e status.

Utilize o comando cryptsetup na partição /dev/sdc1:

```
1 # cryptsetup -y --cipher aes-cbc-essiv:sha256 --key-size 256
     luksFormat /dev/sdb1
2
3 WARNING!
4 ========================================================
5 This will overwrite data on /dev/sdc1 irrevocably.
6
7 Are you sure? (Type uppercase yes): YES (Digite YES em maiúscula)
8 Enter LUKS passphrase: (Digite uma frase secreta)
9 Verify passphrase: (Repita a frase secreta)
```

Descrição dos parâmetros:

-y ou –verify-passphrase: Faz a checagem da senha digitada;

–cipher: Define qual é o modo de criptografia que o dispositivo vai usar;

–key-size: Define em bits o tamanho da chave de criptografia;

LuksFormat: Indica que o dispositivo vai utilizar o padrão LUKS.

Nesse momento outros módulos de criptografia foram ativados:

```
1 # lsmod | grep aes
2 aes_i586                6816  0
3 aes_generic            25738  1 aes_i586
```

14.1.7 Criar mapeamentos para partiçoes LUKS

Agora que aplicamos o padrão LUKS na partição, podemos criar um mapeamento Device-Mapper através do opção luksOpen. Sera solicitada a frase secreta para continuar:

```
1 # cryptsetup luksOpen /dev/sdc1 cryptfs
2 Enter passphrase for /dev/sdc1: (Digite uma frase secreta)
```

Use o comando blkid para identificar o nossa partição como um dispositivo do tipo “crypt_LUKS”.

```
1 # blkid | grep sdc1
2 /dev/sdc1: UUID="fd0e5672-b321-4724-8285-3a30962af074" TYPE="
    crypto_LUKS"
```

Com a opção luksOpen nosso dispositivo /dev/sdc1 foi mapeado para /dev/mapper/-cryptfs, e a partir de agora só pode ser acessado por por este caminho!

14.1.8 Aplicar sistema de arquivos e Montagem

Antes de montar a partição criptografada é preciso aplicar um sistema de arquivos através do comando mkfs:

```
1 # mkfs -t ext4 /dev/mapper/cryptfs
```

Faça a montagem da partição através do comando mount:

```
1 # mkdir /mnt/seguro
2 # mount -t ext4 /dev/mapper/cryptfs /mnt/seguro
```

O que é preciso para montar partições criptografadas durante o Boot?

Quando estamos manipulando partições criptografadas não basta apenas acrescentar a partição ao /etc/fstab, é preciso configurar mais um arquivo, o /etc/crypttab.

```
1 # vim /etc/crypttab
2 <target name>      <source device>      <key file>      <options>
3 cryptfs              /dev/sdc1             none             luks
```

Descrição dos parâmetros:

<target name>: Define o nome que a partição sera mapeada;

<source device>: Define o nome real da partição;

<key file>: Define a frase secreta, use none para nenhum;

<options>: Ativa o dispositvo com extensões LUKS.

Não esqueça de configurar uma nova entrada no arquivo /etc/fstab:

```
# vim /etc/fstab
/dev/mapper/cryptfs  /mnt/seguro  ext4   defaults  0   2
```

Testando o funcionamento:

Para testar o funcionamento sem precisar reiniciar desmonte a partição:

```
# umount /mnt/seguro
```

E desative o dispositivo criptografado

```
# cryptsetup luksClose cryptfs
```

Inicie o script cryptdisks e sera solicitado a frase secreta.

```
# /etc/init.d/cryptdisks start
Starting remaining crypto disks...cryptfs (starting)...
Unlocking the disk /dev/sdc1 (cryptfs)
Enter passphrase: (Digite uma frase secreta)
cryptfs (started)...done
```

Toda vez que o computador for iniciado ou este disco montando em outras maquinas, sera solicitada a frase secreta!

14.1.9 Montagem manual de partições criptografas

Como desativar as partições criptografadas durante o boot?

A ativação automática durante o boot em um servidor não é uma boa pratica de administração, pois acaba atrasando a inicialização do serviços. A solução é alterar a opção “NO” para “YES” da diretiva “CRYPTDISKS_ENABLE” no arquivo /etc/default/-cryptdisks:

```
1 # vim /etc/default/cryptdisks
2 CRYPTDISKS_ENABLE=NO
```

Para ativar de forma manual é só usar o comando cryptdisks_start e o nome mapeado da partição. Mas ainda não vai funcionar porque como desativamos o cryptdisks do boot, o modulo “dm_crypt” não sera carregado. Este módulo é responsável em fazer o mapeamento de dispositivos criptografados.

Vamos resolver este problema ativando o módulo no boot com o comando:

```
1 # echo dm_crypt >> /etc/modules
```

Pronto agora você pode montar a partição com o comando cryptdisks_start!

```
1 # cryptdisks_start cryptfs
2 Starting crypto disk...cryptfs (starting)...
3 Unlocking the disk /dev/sdc1 (cryptfs)
4 Enter passphrase: (Digite uma frase secreta)
5 cryptfs (started)...done
```

Para desativar use o comando cryptdisks_stop e o nome mapeado!

14.1.10 Configurar automount para partições comuns

Como configurar automount no sistema?

Esta pratica pode ser feita através do pacote autofs5, usado para controlar a operação dos "daemons"de auto montagem. Os "daemons"de auto montagem automaticamente montam sistemas de arquivos quando eles são usados e desmontados após um período de inatividade.

Para começar instale o pacote autofs5

```
1 # aptitude install autofs5 -y
```

Sera criado o arquivo /etc/auto.master que é consultado para configurar o automount e os pontos de montagem, gerenciados quando o script autofs é invocado, ou o programa de montagem quando executado de forma automática.

Para testar o automount comece criando uma configuração para o ponto de montagem /dexter/montagem/banco:

```
1 # vim /etc/auto.master
2 /dexter/montagem  /etc/auto.banco   --timeout=10
```

Descrição das opções:

/dexter/montagem: Define em qual diretório sera executado o automout;

/etc/auto.banco: Qual arquivo sera configurado as opções de montagem;

–timeout: Tempo de desmontagem automática. Executado quando nenhum processo ou usuário estiver usando o diretório.

Agora vamos criar o arquivo com as opções de montagem:

```
1 # vim /etc/auto.banco
2 banco -fstype=xfs,rw  :/dev/sdc1
```

Descrição das opções:

banco: Define qual diretório sera montado em /dexter/montagem de forma automática;

-fstype: Define as opção de montagem da partição;

:/dev/sdc1: Define o dispositivo que sera montado.

Para testar remova a entrada no /etc/fstab e reinicie o serviço autofs.

```
1 # vim /etc/fstab
2 # /etc/init.d/autofs restart
```

Para testar acesse o diretório /dexter/montagem/bancoo e use o comando df para listar a partição montada de forma automática:

```
1 # cd /dexter/montagem/banco
2 # df -Th
```

14.1.11 Criar automount para partições criptografadas

Como configurar automount para as partições criptografadas?

Comece criando uma configuração para o ponto de montagem /mnt/seguro:

```
1 # vim /etc/auto.master
2 /mnt  /etc/auto.crypt   --timeout=10
```

Descrição das opções:

/mnt: Define em qual diretório sera executado o automout;

/etc/auto.crypt: Qual arquivo sera configurado as opções de montagem;

–timeout: Tempo de desmontagem automática. Executado quando nenhum processo ou usuário estiver usando o diretório.

Agora vamos criar o arquivo com as opções de montagem:

```
1 # vim /etc/auto.crypt
2 seguro  -fstype=ext4,rw :/dev/mapper/cryptfs
```

Descrição das opções:

seguro: Define qual diretório sera montado em /media de forma automática;

-fstype: Define as opção de montagem da partição;

:/dev/mapper/cryptfs: Define a partição criptograda que sera montada.

Para testar remova a entrada no /etc/fstab e reinicie o serviço autofs.

```
1 # vim /etc/fstab
2 ##/dev/mapper/cryptfs        /mnt/seguro       ext4     noauto,
     defaults     0   0
3 # /etc/init.d/autofs restart
```

Para testar acesse o diretório /mnt/seguro e use o comando df para listar a partição montada de forma automática:

```
1 # cd /mnt/seguro
2 # df -Th
```

455

Linux Essentials

www.4linux.com.br

Conteúdo

Inicialização do Sistema

15.1 Introdução Teórica

Para que possamos entender a base de funcionamento do sistema de inicialização padrão System V, precisamos antes conhecer um tipo especial de arquivos: os links. Um Link serve para termos o mesmo arquivo em diversos lugares, sem a necessidade de cópias. Isso faz com que você tenha a maior quantidade de arquivos em vários lugares e ocupando menos bits de metadados ao sistema de arquivos. O diretório /etc/init.d por exemplo, guarda os scripts para iniciar, e parar determinados serviços. E dentro do diretório rcN.d (a letra N é correspondente ao nível de inicialização, veremos a seguir) que ficam os links dos scripts que estão dentro de /etc/init.d para poder determinar qual script será executado primeiro.

Os links nos permitem fazer modificações nos arquivos originais, assim não precisamos alterar o arquivo original e sua cópia.

15.1.1 System V

O Padrão System V define, entre outras coisas, como deve ser a inicialização dos serviços do sistema. Ele trabalha com níveis de inicialização, os chamados “runlevels”, havendo oito deles que serão descritos posteriormente.

A inicialização do sistema se inicia com um boot loader no qual o usuário escolhe qual sistema operacional ele irá iniciar na máquina. Uma vez escolhido, o boot loader

inicia o carregamento do kernel na memória RAM e passa o controle do sistema a ele. Uma vez que o kernel já esteja controlando a máquina, é iniciada a fase de subir os serviços necessários para a utilização do sistema. Este último estágio é que trataremos aqui.

15.1.2 Níveis de Execução - System V

Em um sistema padrão System V, existem oito níveis de inicialização, sendo eles:

Nível de Execução	Descrição
0	**Shutdown** – Nunca deve ser usado como padrão, porque senão o sistema nunca se inicializará, uma vez que depois do init, ele viria pedindo para desligar.
1,s,S	Modo de **usuário único**, também conhecido como modo de manutenção. Neste modo os serviços do sistema tais como interfaces de rede,servidores web e compartilhamento de arquivos não rodam.
2	**Multiusuário** : No **Debian** é o nível **padrão**, no **RedHat** é um modo **multiuser** sem compartilhamento de arquivos NFS
3	No Red Hat é o **modo multiusuário padrão**. Este e os níveis 4, e 5 não são usados no Debian
4	Normalmente não é utilizado.
5	No RedHat **mutliusuário completo** com login **GUI**, é semelhante ao mod 3, porém o X11 também sobe possibilitando um login GUI.
6	**Restart**, da mesma maneira que o nível 0 , não deve ser usado como padrão, pois o restart será um loop.

Para saber o nível em que se encontra:

```
# runlevel
N 2
```

Onde:

N - nível anterior, quando “N” significa que não houve mudança de nível desde a inicialização.

2 - nível atual.

Para trocar de nível:

```
1 # init <nível>
```

Ex: Para trocar para o nível 3

```
1 # init 3
```

Verifique que o nível de inicialização foi trocado:

```
1 # runlevel
2 2 3
```

Trocar para o modo mono-usuário:

```
1 # init 1
```

Verifique que ao trocar para o modo mono-usuário é executado o nível S, serviços essenciais:

```
1 # runlevel
2 1 S
```

A maioria das distribuições Linux utilizam o padrão System V para gerenciamento dos Daemons e serviços que devem ou não ser carregados nos diferentes níveis de execução.

Para uma melhor compreensão, é interessante que analisemos o arquivo /etc/inittab para ver como esse gerenciamento funciona.

```
1 # vim /etc/inittab
```

Os primeiros serviços a serem inicializados são aqueles do **nível S** que carregam por exemplo o hostname da máquina (serviço essencial). Após esse nível ter sido concluído passa-se para o nível seguinte definido como padrão do sistema no arquivo **/etc/inittab.** Que no caso do **Debian é o nível 2,** e no CentOS é o nível 5. Neste nível são iniciados os outros serviços não essenciais, como servidores de SSH, web, etc.

O sistema System V consiste em agrupar todos os scripts de inicialização do sistema em um único diretório **/etc/init.d** e criar links simbólicos para esses scripts dentro dos diretórios dos runlevels apropriados. Cada runlevel possui o seu diretório, sendo eles localizados no /etc sobre o nome rcN.d, no qual o caracter **'N'** representa o número do runlevel.

Nesses diretórios haverá links para os scripts de inicialização e/ou finalização dos serviços e o nome desses links indicará se o serviço deve ser iniciado ou finalizado e qual será a ordem que será seguida para isso.

Por exemplo, se um serviço começa por S18 como é o caso do ssh cujo nome dentro do rc2.d é **S18ssh;** ele será o serviço a ser **iniciado** após todos os serviços com número menor que o dele serem iniciados. Por exemplo, ele será iniciado após o serviço “portmap” cujo nome do link é **S14portmap**, caso exista outro serviço com o mesmo número de inicialização do ssh, como o agendador de tarefas “at” cujo nome do link é S18atd, a ordem de inicialização se dará pela ordem alfabética, ou seja o S18atd será executado antes de S18ssh.

No caso de um desses links ter seu nome iniciando pela letra **K** esse serviço será finalizado quando o runlevel correspondente for iniciado. Dessa forma se existir um link chamado **K01atd** no runlevel 0 (/etc/rc0.d), quando mudarmos para esse runlevel, se o atd estiver ativo ele será o um dos primeiros a ser desativado.

15.1.3 O que faz um script de inicialização?

Um script de inicialização nada mais é do que um script que realiza verificações essenciais ao funcionamento do serviço em questão e uma estrutura do tipo "case" que aceitará os argumentos start|stop|restart dentre outros. Sendo assim, para iniciar um serviço basta dar o comando:

```
1 # /etc/init.d/<nome_do_serviço> start
```

E para finalizar um serviço basta executar com o parâmetro stop:

```
1 # /etc/init.d/<nome_do_serviço> stop
```

EX: Parando o serviço do ssh:

```
1 # /etc/init.d/ssh stop
```

Ex: Inicializando o serviço do ssh:

```
1 # /etc/init.d/ssh start
```

No Debian 6.0 todos os scripts de inicialização "/etc/init.d/"foram convertidos para ordenar a sequência de boot baseado em um padrão especificado na Linux Standard

Base (LSB). Com a adesão deste padrão agora existe um cabeçalho em todos os scripts de inicialização onde são indicadas as dependências que ele necessita para poder ser executado, fazendo com que o script inicie só depois de tais dependências.

Esta funcionalidade é ativada pelo comando "insserv" que ordena os scripts init.d baseando-se nas suas dependências declaradas nos cabeçalhos. Ou seja, para adicionar/remover serviços da inicialização no Debian 6.0 não se usa mais o "update-rc.d"e sim o "insserv"!

Vamos usar o comando head que mostra por padrão as 10 primeiras linhas e verificar o cabeçalho de inicialização do ssh:

```
# head /etc/init.d/ssh
#! /bin/sh

### BEGIN INIT INFO
# Provides:         sshd
# Required-Start:   $remote_fs    $syslog
# Required-Stop:    $remote_fs    $syslog
# Default-Start:    2 3 4 5
# Default-Stop:
# Short-Description:   OpenBSD Secure Shell server
### END INIT INFO
```

O comando **head**, por padrão, exibe as 10 primeiras linhas de um arquivo.

A opção mais usada é:

- Provides: Nome do script

- Required-Start: Defini scripts que devem ser inicializados antes de carregar este script.

- Required-Stop: Defini scripts que devem ser parados, antes de descarregar este script.

- Should-Start: Define que irá rodar só depois que os serviços declarados forem inicializados.

- Should-Stop: Defini que irá parar só depois que os serviços declarados forem parados.

- Default-Start: Níveis para carregar o serviço

- Default-Stop: Níveis para descarregar o serviço

- Short-Description: Descrição rápido do script

- Description: Descrição mais detalhada do script

Valores genéricos:

```
1  $local_fs
2    Todos os arquivos locais de sistema são montados..
3  $network
4    Baixo nível de rede. (placa de rede; PCMCIA)
5  $named
6    Daemons que podem fornecer resolução de nomes, como por exemplo:
       DNS, NIS, LDAP.
7  $portmap
8    Daemon que fornece mapeamento de portas.
9  $remote_fs
10   Todos arquivos de sistema estão montados.
11 $syslog
12   Logs do sistema operacional.
13 $time
14   Daemons utilizados para definir a hora do sistema, como ntpdate,
       ntp, rdate.
```

```
15 $all
```

Suportado pelo insserv para iniciar depois que todos os outros scripts forem carregados na sequência de inicialização. Somente trabalha para inicializar, para descarregar não é possível, pois nenhum script depende de todos.

15.1.4 Removendo um Script da Inicialização

Debian:

Removendo o ssh da inicialização do sistema:

```
1 # insserv -rv ssh
2 insserv: remove service /etc/init.d/../rc2.d/S18ssh
3 insserv: remove service /etc/init.d/../rc3.d/S18ssh
4 insserv: remove service /etc/init.d/../rc4.d/S18ssh
5 insserv: remove service /etc/init.d/../rc5.d/S18ssh
6 insserv: creating .depend.boot
7 insserv: creating .depend.start
8 insserv: creating .depend.stop
```

Adicionando o ssh na inicialização do sistema:

```
1 # insserv -v ssh
2 insserv: enable service ../init.d/cron -> /etc/init.d/../rc2.d/
     S18ssh
3 insserv: enable service ../init.d/cron -> /etc/init.d/../rc3.d/
     S18ssh
4 insserv: enable service ../init.d/cron -> /etc/init.d/../rc4.d/
     S18ssh
5 insserv: enable service ../init.d/cron -> /etc/init.d/../rc5.d/
     S18ssh
```

```
6 insserv: creating .depend.boot
7 insserv: creating .depend.start
8 insserv: creating .depend.stop
```

E de onde o insserv tira a prioridade de rodar os scripts? Exemplo do ssh ("/etc/init.d/../rc2.d/S18ssh") iniciando com prioridade 18? Aí que está a grande novidade, você não precisará mais editar essa prioridade na mão, o insserv vai calcular a prioridade de acordo com o tal cabeçalho verificando quais os scripts que serão iniciados antes para que o script seja executado.

red hat

Red Hat: Nível 0 desliga o sistema; Nível 1 modo mono usuário; Nível 2 multiusuário, sem NFS; Nível 3 multi-usuário, com NFS, sem X; Nível 4 não usado; Nível 5 multi-usuário com NFS e X; Nível 6 reinicializar o sistema;

No Red Hat os serviços ficam em **/etc/rc.d/init.d,** todos os arquivos aqui têm um hard link para **/etc/init.d.** Então tanto faz mudar em um como em outro, ambos serão atualizados.

Vamos olhar o cabeçalho de inicialização do sshd do CentOS:

```
1 # head -n 15 /etc/rc.d/init.d/sshd
2 #!/bin/bash
3 #
4 # Init file for OpenSSH server daemon
5 #
6 # chkconfig: 2345 55 25
7 # description: OpenSSH server daemon
8 #
9 # processname: sshd
10 # config: /etc/ssh/ssh_host_key
```

```
11 # config: /etc/ssh/ssh_host_key.pub
12 # config: /etc/ssh/ssh_random_seed
13 # config: /etc/ssh/sshd_config
14 # pidfile: /var/run/sshd.pid
```

Onde:

```
1 # chkconfig:
```

<Níveis de execução para inicialização> 2345 <ordem iniciar ”S”> S55 <ordem parar ”K”> K25

- description: Descrição

- config Arquivos de configuração.

- pidfile Localização do PID do processo.

15.1.5 Gerenciando Serviços

Para verificar os serviços habilitados em cada runlevel digite:

```
1 # chkconfig --list
```

Para serviço específico:

```
1 # chkconfig --list httpd
```

Adicionar serviço ssh na inicialização:

```
1 # chkconfig --add sshd
```

Para desabilitar serviço do ssh da inicialização:

```
1 # chkconfig --del sshd
```

Ou:

```
1 # chkconfig sshd off
```

Nota: Estas linhas garantem que o serviço do ssh esteja desabilitado no próximo reboot. Para desligar o mesmo serviço sem precisar reiniciar a máquina use o comando:

```
1 # /etc/init.d/sshd stop
```

Se não desligares o serviço com chkconfig, ele será reinicializado da próxima vez que o computador for reiniciado, mesmo que o tenhas parado através do script apropriado (notar que 'stop' é um argumento de entrada para o mesmo script). Por isso para parar imediatamente e desativar permanentemente o serviço deves usar:

```
1 # chkconfig --del sshd  && /etc/init.d/sshd stop
```

Ou:

```
1 # chkconfig sshd off && /etc/init.d/sshd stop
```

Para habilitar o serviço ssh:

```
1 # chkconfig sshd on
```

Ou:

```
1 # chkconfig --add sshd
```

Para habilitar o serviço ssh em determinados níveis:

```
1 # chkconfig --level 23 sshd on
```

Foi habilitado o ssh nos níveis 2 e 3, visualize:

```
1 # chkconfig --list sshd
```

Para desabilitar o serviço ssh em determinados níveis:

```
1 # chkconfig --level 2 sshd off
```

Foi desabilitado o ssh no nível 2 , visualize:

```
1 # chkconfig --list sshd
```

455

Linux Essentials

www.4linux.com.br

Conteúdo

0.1 Introdução ao Upstart

O Upstart é um sistema de inicialização de serviços diferente do System V. Sendo o substituto do sysvinit, o Upstart ainda consegue manter a compatibilidade com os scripts do sistema. Seu funcionamento se baseia em eventos para tomar decisões ao funcionamento do sistema, como por exemplo, monitorar serviços quando iniciam, quando param ou apresentam algum erro. No Upstart a comunicação dos serviços

com o init ocorre por meio do D-Bus.

O Upstart comparado ao SysV init é mais rápido porque consegue executar os scripts de inicialização de forma paralela, dependendo que eventos sejam definidos na inicialização do sistema. Uma outra vantagem é que o Upstart pode monitorar um serviço, e em caso de interrupção o mesmo é inicializado de forma automática.

0.2 Instalação

A instalação é dispensada dependendo da distribuição. Em distribuições baseadas em Debian é possível usar o comando:

```
1 # aptitude install upstart
```

Para ver na prática o Upstart vamos utilizar a distribuição CentOS 6

Mas quais distribuições usam o Upstart?

- Ubuntu 9.10: Upstart nativo;
- Fedora 9: Substituiu o sysvinit pelo Upstart apartir dessa versão;
- Fedora 15: O Upstart foi substituído por Systemd nesta versão;
- RHEL 6 e CentOS 6: Inclui o Upstart;
- Debian: Planos para migrar para o Systemd em versões futuras;
- OpenSUSE: Incluiu o Upstart na versão 11,3 mas não como padrão;
- NixOS: Utilizando o Upstart como padrão;

- Maemo 5: Upstart substitui o sysvinit no sistema operacional para Internet
- Google OS: Utiliza o Upstart;

O Upstart ainda é usado no webOS da HP para o Palm Pre, Palm Pixi (ambos antes de a Palm foi comprada pela HP), HP Veer, e HP Pre 3 telefones inteligentes, juntamente com o tablet TouchPad HP.

0.3 Diretórios e arquivos de configuração

Vamos começar com o diretório /etc/init:

```
1 # ls -l /etc/init
```

O diretório /etc/init possui arquivos de serviços do Upstart (arquivos que definem os tarefas que o daemon Upstart init é executado). Inicialmente, este diretório é preenchido pelo pacote de software Upstart. Em algumas versões do Ubuntu o nome do diretório é /etc/event.d.

Quando você instala novos serviços, serão acrescentados novos arquivos neste diretório para controlar o serviço instalado, substituindo os arquivos que foram previamente colocados no diretórios /etc/rc.d/rc* e /etc/rc.d/init.d.

Para aprender um pouco mais sobre o Upstart abra um arquivo .conf do diretório /etc/init.

```
1 # vim /etc/init/control-alt-delete.conf
2 # control-alt-delete - emergency keypress handling
3 # This task is run whenever the Control-Alt-Delete key combination
    is
4 # pressed.  Usually used to shut down the machine.
```

```
5 start on control-alt-delete
6 exec /sbin/shutdown -r now "Control-Alt-Delete pressed"
```

Em nosso exemplo temos o arquivo que configura o evento CTRL + AL +DEL. Acompanhe a seguir a descrição das linhas:

- start on: Define em qual evento o arquivo sera lido. Em nosso exemplo toda vez que for acionado a combinação de teclas CTRL + ALT +DEL, a próxima linha sera lida;
- exec: Indica qual comando ou script sera executado.

Devido a migração do Sysvinit para o Upstart, só é possível desativar o CTRL + ALT + DEL no arquivo /etc/init/control-alt-delete.conf e não mais no /etc/init/inittab!

O que é um evento?

Um evento é um argumento que envolve o carregamento do sistema, a montagem do sistema de arquivos, combinação de teclas, execução de um ou mais runlevels, a execução de um script em /etc/init.d ou até mesmo quando uma interfaces de rede estiver UP. Exemplos:

- carregamento do sistema: start on startup
- montagem do sistema de arquivos: start on filesystem
- combinação de teclas: start on control-alt-delete
- execução nos runlevels 2 a 5: start on runlevel [2345]
- execução nos runlevels S, 1, 0 e 6: stop on runlevel [S016]
- execução de um script em /etc/init.d: start on started messagebus

- interfaces de rede estiver UP: start on net-device-up IFACE=lo

Para combinar dois eventos ao mesmo tempo use “and” entre as opções:

start on filesystem and net-device-up IFACE=lo

0.4 Como é possível chamar um evento?

Os eventos de teclas é só acionar no teclado, de runlevel use o comando init, startup e filesystem quando iniciar o sistema, de interface quando a mesma estiver UP e de scripts do /etc/init.d na inicialização do sistema.

Comando initctl

O comando initctl permite se comunicar com o daemon do Upstart e gerenciar os scripts localizados em /etc/init. Veja na pratica os exemplos:

Listar o estado atual de todos os scripts do Upstart

```
1  # initctl list
2  rc stop/waiting
3  tty (/dev/tty3) start/running, process 1662
4  tty (/dev/tty2) start/running, process 1660
5  tty (/dev/tty6) start/running, process 1674
6  tty (/dev/tty5) start/running, process 1666
7  tty (/dev/tty4) start/running, process 1664
8  plymouth-shutdown stop/waiting
9  control-alt-delete stop/waiting
10 rcS-emergency stop/waiting
11 readahead-collector stop/waiting
12 kexec-disable stop/waiting
13 rcS stop/waiting
14 prefdm start/running, process 3997
```

```
init-system-dbus stop/waiting
readahead stop/waiting
splash-manager stop/waiting
start-ttys stop/waiting
readahead-disable-services stop/waiting
rcS-sulogin stop/waiting
serial stop/waiting
```

Listar apenas o estado atual de um script

```
# initctl status control-alt-delete
```

Chamar um evento por linha de comando

```
# initctl emit control-alt-delete
```

0.5 Personalizar novos scripts

Como exemplo vamos criar um script que ira gravar a data e hora de 3 em 3 segundos em um arquivo no diretório do root.

Primeiro crie o script no diretório do /root

```
# vim /root/script
#!/bin/bash
while true
do date >> /root/data.txt
sleep 3
done
```

Torne o script executável através do comando chmod

```
# chmod u+x /root/script
```

Agora crie o script Upstart de nome date.conf no diretório etc/init

```
# vim /etc/init/date.conf

start on runlevel [2345]
stop on runlevel [S016]
respawn
exec /root/script
```

Descrição das opções:

- start on runlevel [2345]: Inicia a execução do script nos runlevels 2 a 5
- stop on runlevel [S016]: Para a execução do script nos runlevels S, 1, 0 e 6
- respawn: Significa que o processo será reiniciado se terminar inesperadamente.
- exec: Indica qual comando ou script sera executado.

Agora inici o script através do comando start

```
# start date
date start/running, process 6219
```

O resultado o comando é o estado do script e seu PID.

Mostre o estado do script através do comando status

```
# status date
date start/running, process 6219
```

Pare a execução do script através do comando stop

```
# stop date
date stop/waiting
```

Agora para testar a opção “respawn” inicie o script date e use o comando kill -9 PID do mesmo. Ao usar comando status o script volta a se executado de forma automática.

```
# start date
date start/running, process 6317
```

```
# kill -9 6317
# status date
date start/running, process 6340
```

0.6 Introdução ao Systemd

O Systemd é um gerenciador de inicialização e serviços sendo o futuro substituto do SystemVinit, fornecendo recursos de paralelização agressivos com vários serviços iniciando ao mesmo tempo.

Oferece supervisão de processos utilizando cgroups e a capacidade de não só depender, de outro script de inicialização que está sendo iniciado, mas também a disponibilidade de um ponto de montagem ou serviço Dbus. Tem suporte a snapshotting

e restauração do estado do sistema, alem de manter montagem e pontos de auto-mount sobre demanda.

Quais são as vantagens em migrar para o Systemd?

- Boot mais rápido: O Systemd consegue iniciar menos processos no total e iniciar mais processos em paralelo durante o boot;

- Montagem sobre demanda: O Systemd usa o automont para controlar a montagem do sistema de arquivos;

- Grupo de processos: O uso de cgroups agrupa processos pai, filhos e netos em um único grupo, facilitando o gerenciamento de recursos em memoria e processamento;

0.7 Instalação

A instalação é dispensada dependendo da distribuição. Em distribuições baseadas em Debian é possível usar o comando:

```
1 # aptitude install systemd
```

Para ver na prática o Systemd vamos utilizar a distribuição Fedora 15

i386: http://fedora.c3sl.ufpr.br/linux/releases/15/Fedora/i386/iso/Fedora-15-i386-DVD.iso

x86_64: http://fedora.c3sl.ufpr.br/linux/releases/15/Fedora/x86_64/iso/Fedora-15-x86_64-DVD.iso

0.8 Diretórios e arquivos de configuração

Vamos começar com o binário do SystemD:

```
1 # ls -l /sbin/init
2 lrwxrwxrwx. 1 root root 14 Abr 24 16:52 /sbin/init -> ../bin/systemd
```

O sistema cria um link de nome init para carregar durante a inicialização, mas na verdade que é carregado é o systemd.

0.9 Unidade de Serviços

O Systemd armazena suas configurações de serviços no diretório /usr/lib/system. O daemon Systemd init é baseado no conceito de unidades e cada uma das quais tem um nome e um tipo, e seu controle é feito pelo arquivo dependendo de sua extensão.

Os tipos de unidades são classificadas como Serviço (.service), Socket (.socket), Dispositivo de montagem (.mount), Automount (.automount), Alvo (.target), Instantâneo (.snapshot), Tempo (.timer), Swap (.swap) e Caminho (.path).

Mas o que é uma unidade de serviço?

A unidade de serviço refere-se a um daemon (serviço) que o systemd pode controlar, incluindo scripts controlados nativamente pelo Systemd e aqueles controlados por Systemd via scripts Sysvinit. Por exemplo, o Systemd controla o daemon ntpd nativamente através da unidade de serviço ntpd.service.

Vamos analisar o conteúdo do arquivo para aprender um pouco mais

```
1 # cat /usr/lib/system/ntpd.service
2 [Unit]
3 Description=Network Time Service
4 After=syslog.target ntpdate.service
5
6 [Service]
7 EnvironmentFile=/etc/sysconfig/ntpd
8 ExecStart=/usr/sbin/ntpd -n -u ntp:ntp $OPTIONS
```

O Arquivo possu duas sessoes (stanza):

- Unit: Indica a descrição da unidade do serviço (Description) e After define as suas dependências.
- Service: Indica o que sera executado (ExecStart) quando a unidade de serviço ntpd.service for carregado no boot ou por outro comando. A opção “EnvironmentFile” define as configurações para o serviço, como por exemplo a variavel OPTIONS que sera definida na opção ExecStart.

0.10 Gerenciando serviços

Para controlar o sistema systemd e gerenciar as unidades de serviços use o comando systemctl. Vamos ver na pratica alguns exemplos de utilização:

Para trazer informações sobre a unidade de serviço ntp use a opção status:

```
# systemctl status ntpd.service
   ntpd.service - Network Time Service
   Loaded: loaded (/lib/systemd/system/ntpd.service)
   Active: inactive (dead)
   CGroup: name=systemd:/system/ntpd.service
```

O comando informa a descrição, qual o caminho do arquivo de configuração, se o serviço esta ativo e quais são os grupos de processos em Cgroup.

Para iniciar o serviço use a opção start:

```
# systemctl start ntpd.service
```

Use novamente a opção status para ver a diferença:

```
# systemctl status ntpd.service
ntpd.service - Network Time Service
   Loaded: loaded (/lib/systemd/system/ntpd.service)
   Active: active (running) since Tue, 24 Apr 2012 18:32:52 -0300;
      57s ago
   Main PID: 24647 (ntpd)
   CGroup: name=systemd:/system/ntpd.service
                 24647 /usr/sbin/ntpd -n -u ntp:ntp -g
           24648 /usr/sbin/ntpd -n -u ntp:ntp -g
```

O que temos de novo é o estado (running) a data e hora que foi ativado e o PID

Para reiniciar o serviço use a opção restart:

```
# systemctl restart ntpd.service
```

Para parar o serviço use a opção stop:

```
1 # systemctl stop ntpd.service
```

Use as opções disable e enabled para desativar ou ativar um serviço no boot:

```
1 # systemctl disable ntpd.service
2 rm '/etc/systemd/system/multi-user.target.wants/ntpd.service'
3
4 # systemctl enable ntpd.service
5 ln -s '/lib/systemd/system/ntpd.service' '/etc/systemd/system/multi-
    user.target.wants/ntpd.service'
```

E para verificar se o serviço já esta ativado no boot use a is-enabled

```
1 # systemctl is-enable ntpd.service ; echo $?
```

Mas em qual runlevel o serviço do ntpd sera iniciado?

O Systemd dispensa o conceito de runlevels, mas mantém uma compatibilidade através dos diretórios /lib/systemd/system/runlevel*.target:

```
1 # ls -l /lib/systemd/system/runlevel*.target
2 /lib/systemd/system/runlevel0.target -> poweroff.target
3 /lib/systemd/system/runlevel1.target -> rescue.target
4 /lib/systemd/system/runlevel2.target -> multi-user.target
5 /lib/systemd/system/runlevel3.target -> multi-user.target
6 /lib/systemd/system/runlevel4.target -> multi-user.target
7 /lib/systemd/system/runlevel5.target -> graphical.target
8 /lib/systemd/system/runlevel6.target -> reboot.target
```

Isso quer dizer que quando um serviço é ativado ele ganha um link simbólico no diretório /etc/systemd/system/multi-user.target.wants/, como foi o caso do ntpd.service!

Como definir o runlevel padrão no Systemd?

Quando usamos o SysVinit a alteração era feita através do arquivo /etc/inittab, mas no Systemd essa alteração é feita através de arquivos .target e links. Exemplo:

Para exibir qual é o runlevel padrão do sistema use o comando:

```
1 # ls -l /etc/systemd/system/default.target
2
3 /etc/systemd/system/default.target -> /lib/systemd/system/runlevel5.
    target
```

O comando retorna que o runlevel padrão aponta para o arquivo default.traget, que aponta para o graphical.target que corresponde ao modo gráfico no Systemd.

```
1 # ls -l /lib/systemd/system/runlevel5.target
2
3 /lib/systemd/system/runlevel5.target -> graphical.target
```

Para definir o modo texto como padrão de inicialização, o default.target deve ser um link apontando para o arquivo multi-user.target. Veja no exemplo:

```
1 # ln -sf /lib/systemd/system/multi-user.target /etc/systemd/system/
    default.target
```

Ainda é possível verificar qual o modo de inicialização esta definido (gráfico ou texto) através do comando systemctl com a opção “list-units”

```
1 # systemctl list-units --type=target | grep user
2 multi-user.target          loaded active active     Multi-User
```

Agora toda vez que a maquina for iniciada o modo de execução sera em texto e multiusuário. Para alternar para o modo gráfico use a opção "isolate" através do comando systemctl.

```
1 # systemctl isolate grapical.target
```

0.11 Personalizar novos scripts

Para criar novos arquivos de unidades nunca crie no diretório /lib/systemd/system e em /etc/systemd/system. Como exemplo pratico vamos criar um script na iniciar o script que gava de 3 em 3 segundos a data e hora no arquivo data.txt

Comece criando o arquivo de unidade de serviço

```
1  # vim /etc/systemd/system/date.service
2
3  [Unit]
4  Description=Date Daemon
5
6  [Service]
7  ExecStart=/root/script
8  ExecStop=/bin/kill -TERM $MAINPID
9
10 [Install]
11 WantedBy=multi-user.target
```

Detalhando as novas opções:

- ExecStop: Defina qual comando ira encerar o serviço
- Install: Cadastra a unidade no target multi.user do Systemd

Para visualizar o status use o comando systemctl com a opção status

```
1 # systemctl status date.service
2
3 date.service - Date Daemon
4    Loaded: loaded (/etc/systemd/system/date.service)
5    Active: inactive (dead)
6    CGroup: name=systemd:/system/date.service
```

Inicie o serviço e verifique novamente seu status

```
1 # systemctl start date.service
2 # systemctl status date.service
3
4 date.service - Date Daemon
5    Loaded: loaded (/etc/systemd/system/date.service)
6    Active: active (running) since Tue, 24 Apr 2012 20:03:09 -0300; 1
      s ago
7    Main PID: 1495 (script)
8    CGroup: name=systemd:/system/date.service
9                 1495 /bin/bash /root/script
10          1497 sleep 3
```

Para terminar encerre o serviço com o comando systemctl

```
1 # systemctl stop date.service
```

E não esqueça de ativar na inicialização do sistema

```
1 # systemctl enable date.service
```

0.12 Troubleshooting

Como posso montar partições sobre demanda no Systemd?

O System não precisa utilizar o arquivo /etc/fstab para realizar montagem, e sim as unidades que recebem a extensão .mount para montagem fixas e .automount para montagem sobre demanda. Em nossa pratica precisamos de uma nova partição ou disco. Em nosso exemplo sera usado uma nova HD /dev/sdb.

Crie uma nova partição através do comando fdisk, aplique o sistema de arquivos EXT4 e crie um ponto de montagem na raiz

Roteiro:

```
1 # fdisk /dev/sdb
2 # mkfs.ext4 /dev/sdb1
3 # mkdir /dados
```

Agora vamos criar os arquivo .mount e .automount no diretório /etc/systemd/system

```
1 # vim /etc/systemd/system/dados.mount
2
3 [Unit]
4 Description=Diretorio Dados
5
6 [Mount]
```

```
7 What=/dev/sdb1
8 Where=/dados
9 Type=ext4
10 Options=noatime
```

As opções para montagem são bem simples de entender. “What” define qual dispositivo sera montado, “Where” o ponto de montagem, “Type” o sistema de arquivos e “Options” opção de montagem.

Agora crie o arquivo automount

```
1 # vim /etc/systemd/system/dados.automount
2
3 [Unit]
4 Description=Montagem automatica do /dados
5
6 [Automount]
7 Where=/dados
```

E não esqueça de ativar na inicialização do sistema

```
1 # systemctl enable dados.automount
```

0.13 Testando a montagem automática

Primeiro verifique o status do automount

```
1 # systemctl status dados.automount
2 dados.automount - Montagem automática do /dados
```

```
3     Loaded: loaded (/etc/systemd/system/dados.automount)
4     Active: inactive (dead)
5     Where: /dados
```

Agora ative o automount com a opção start e veja seu status

```
1  # systemctl start dados.automount
2  # systemctl status dados.automount
3     dados.automount - Montagem automática do /dados
4     Loaded: loaded (/etc/systemd/system/dados.automount)
5     Active: active (waiting) since Tue, 24 Apr 2012 20:39:14 -0300; 1
         s ago
6     Where: /dados
```

O detalhe é que ele esta ativo mais aguardando algum acesso no diretório /dados. Use o comando cd /dados para montar de forma automática, e use a opção status para verificar a diferença.

```
1  # cd /dados
2  # systemctl status dados.automount
3  dados.automount - Montagem automática do /dados
4     Loaded: loaded (/etc/systemd/system/dados.automount)
5     Active: active (running) since Tue, 24 Apr 2012 20:39:14 -0300; 2
         min 55s ago
6     Where: /dados
```

455

Linux Essentials

www.4linux.com.br

Conteúdo

Capítulo 17

Servidor X

17.1 Introdução teórica

O **"X Window System"**, conhecido também como **"servidor X",** apenas **X ou X11**, é um protocolo de rede e vídeo que provê a capacidade de se trabalhar com o sistema de janelas e que permite as interações através de teclado e mouse. Esse sistema fornece os meios para o desenvolvimento de interfaces gráficas para usuários ou **"GUI - Graphical User Interfaces"** em sistemas **"Unix"** e **"Unix-like"**, como o **GNU/Linux**.

O sistema X fornece apenas as ferramentas que possibilitam o desenvolvimento de ambientes "GUI" como desenhar na tela, mover janelas e interagir com o mouse e teclado; ele não dita quais serão as decorações das janelas, quem faz isso são os chamados **"WM - Window Managers"** ou gerenciadores de janelas. Dessa forma, a "cara" da parte gráfica varia drasticamente de um "WM" para outro.

Um conceito básico do servidor X é que ele é realmente um servidor como o próprio nome já indica. Sendo assim, é possível abrir várias instâncias de interface gráfica em uma mesma máquina ou até mesmo em uma máquina remota, graças ao seu protocolo de rede.

17.2 Configurando o suporte à Interface Gráfica

A interface gráfica mais utilizada em ambientes "UNIX" é conhecida como "X Window System" ou simplesmente X. Essa interface é provida pelo pacote "Xorg", que pode ser baixado diretamente no site oficial "http://www.xorg.org" ou utilizando o "aptitude/apt-get"ou "yum" dos pacotes necessários.

Há basicamente quatro formas de configurar o servidor X, sendo elas:

```
1 # X -configure
```

No Debian Lenny 6.0, o X tem uma configuração um pouco menor dado o fato que todas as configurações do "debconf" são aproveitadas para configuração do servidor X;

O arquivo de configuração do servidor X é dividido em seções e cada uma diz respeito à configuração de um determinado pedaço do sistema como um todo. A estrutura básica de um desses arquivos é a seguinte:

- ServerLayout
- InputDevice (mouse)
- Screen
- InputDevice (keyboard)
- Files
- Modules

- InputDevice (mouse)
- InputDevice (keyboard)
- Screen
- Monitor
- Displays
- Device (video card)

Ou seja, o arquivo é composto de várias seções que definem qual será o comportamento dos dispositivos como teclado, mouse, monitor e placa de vídeo e algumas outras, que definem recursos que o servido X irá utilizar, como os módulos que serão carregados e os arquivos de fontes, por exemplo.

Além das seções separadas que definem o comportamento de algum componente extra, há outras como **"ServerLayout"** e "Screen" que definem como o conjunto de recursos irá operar.

17.3 Variável de Ambiente DISPLAY

A variável de ambiente **"DISPLAY"** é a que define em que lugar a saída gráfica deve ser mostrada. Com essa variável definida é possível até informar ao sistema que a saída gráfica se dará em outro computador na rede. O formato de definição dessa variável é o seguinte:

```
1 <ip_destino>:<display>.<screen>
```

Sendo o <ip_destino> o endereço IP de uma máquina na rede, podendo ser deixado em branco caso a máquina de destino seja a própria máquina local. O campo “display” refere-se a uma instância de parte gráfica dentro de uma “screen”. O campo “screen” refere-se ao monitor e à placa de vídeo na qual a parte gráfica será exibida.

Não se esqueça que a variável que define o ambiente do usuário é a “DISPLAY”.

17.4 Window Managers

Um “X Window Manager” é um software que controla basicamente o posicionamento e a aparência das janelas dentro do sistema “X Window”.

Ao contrário dos sistemas da Apple e Microsoft, que possuem apenas uma única aparência básica, e que é de controle delas, nos sistemas GNU/Linux você é livre para escolher qual é o gerenciador de janelas que irá utilizar.

Há um número muito grande de gerenciadores de janelas que você pode instalar simultaneamente em uma máquina, possibilitando que cada usuário escolha aquele que mais lhe agrade. Cada gerenciador difere do outro em muitos aspectos, como nível de customização da aparência e funcionalidades, configuração dos menus, meios gráficos para iniciar um software, capacidade de utilizar múltiplos “desktops” e, principalmente, na quantidade de recursos que ele exige da máquina.

Algumas das opções de gerenciadores são:

AfterSteps	Blakbox	FluxBox
Evilwn	Enlightenment	FVWM
IceWM	Ion	Kwin(KDE)
Metacity (Gnome)	WMN	SawFish
twm	xfce	OpenClasses(Sun)

17.5 Display Managers

Os **"Display Managers"** são programas que agrupam algumas tarefas como realizar a validação do usuário local ou remoto (via protocolo "XDMCP"), além de permitir que o usuário selecione, de forma fácil, qual "Window Manager" ele deseja utilizar.

Alguns exemplos de **"Display Managers"** são o **"KDM"** (padrão do KDE), **"GDM"** (padrão do GNOME) e **"XDM"** (padrão do servidor X).

17.6 Protocolo XDMCP

O **"XDMCP - X Display Manager Control Protocol"** é um protocolo de rede que utiliza a porta "177/udp" e é utilizado para servir interface gráfica para clientes na rede.

Se um "Display Manager" estiver com o protocolo "XDMCP" ativado, basta um servidor X enviar um pacote de "query" à máquina que está servindo o "DM", que esta responderá à máquina solicitante enviando a saída gráfica do "DM" para que algum usuário possa realizar a validação.

Esta é uma forma de utilizar a parte gráfica de outro computador, em uma máquina com menos recursos de hardware, pois o processamento de interface gráfica estará ocorrendo na máquina servidora.

17.7 Xnest

Um **"Xnest"** é uma instância do servidor X que pode ser utilizada para receber alguma saída gráfica que tenha sido redirecionada a ela utilizando a variável "DISPLAY". Pode ser utilizada também para receber um "DM" solicitado via "XDMCP".

17.7.1 Instalação e Configuração do Servidor X

Até a versão "Sarge 3.1" do Debian, o servidor X11 utilizado era o "XFree86", a partir da versão "Etch 4.0", o servidor padrão passou a ser o "Xorg". Agora estamos utilizando a versão "Squeeze 6.0".

Ver informação sobre as janelas:

```
1 # wininfo
```

Ver informações sobre o servidor X:

```
1 # xdpyinfo
```

Qual é o comando que me traz informações sobre as cores e opções do Servidor X? R: "xwininfo"

Pare o servidor x:

```
1 # /etc/init.d gdm3 stop
```

Gere a configuração de vídeo detectada pelo "dexconf":

```
1  # dexconf
```

Caso esteja funcionando, ótimo. De qualquer forma, vamos executar o procedimento de configuração:

```
1  # X -configure
```

Esse comando tentará identificar qual é o hardware da sua máquina e gerar um arquivo de configuração para ela, gravando esse arquivo no diretório do usuário "root".

Teste o novo arquivo de configuração:

```
1  # X -config /root/xorg.conf.new
```

Novamente, se funcionar, ótimo. Caso contrário, teremos que realizar os ajustes manualmente, e para isso precisaremos de algumas informações como:

- placa de vídeo - para determinar qual é a nossa placa de vídeo, execute:

```
1  # lspci | grep -i VGA
```

- frequências do monitor - para descobrir quais são as frequências suportadas pelo seu monitor você deve recorrer ao manual do proprietário ou buscar pelas especificações técnicas em mecanismos de busca disponíveis na Internet.

Vamos visualizar o arquivo de configuração:

```
1 # less /root/xorg.conf
```

Um arquivo de configuração típico:

```
1  Section "InputDevice" (Entrada de Mouse)
2   Identifier "Configured Mouse"
3   Driver "mouse"
4   Option "CorePointer"
5   Option "Device" "/dev/input/mice"
6   Option "Protocol" "ImPS/2"
7   Option "Emulate3Buttons" "true"
8  EndSection
9
10 Section "Device" (Seção que define o nosso hardware de vídeo)
11  Identifier "Video Card"
12  Driver "vesa"
13 EndSection
14
15 Section "Monitor" (Opções de Monitor)
16  Identifier "Generic Monitor"
17  Option "DPMS"
18  HorizSync 28-51
19  VertRefresh43-60
20 EndSection
21
22 Section "Screen" (Layout de Screen e bits de cores a serem
       utilizadas)
23  Identifier "Default Screen"
24  Device "Video Card"
25  Monitor "Generic Monitor"
26  DefaultDepth 24
27  SubSection "Display"
28  Depth 1
```

```
29  Modes "1024x768" "800x600" "640x480"
30  EndSubSection
31 
32 SubSection "Display"
33 Depth 24
34 Modes "1024x768" "800x600" "640x480"
35 EndSubSection
36 EndSection
37 
38 Section "ServerLayout"
39  Identifier "Default Layout"
40  Screen "Default Screen"
41  InputDevice"Generic Keyboard"
42  InputDevice"Configured Mouse"
43 EndSection
```

Leitura sugerida para mais informações a respeito desse arquivo e suas opções de configuração e parâmetros: "man xorg.conf"

Realizadas as alterações, vamos fazer um novo teste para ver se o servidor consegue carregar a parte gráfica.

Teste as configurações:

```
1 # X -config /root/xorg.conf.new
```

Funcionando, basta mover os arquivos para o diretório correto:

```
1 # mv /root/xorg.conf.new /etc/X11/xorg.conf
```

Tente carregar a interface gráfica com os seguintes comandos:

```
# X
Ou:
# startx
```

17.7.2 Instalando um Window Manager

No GNU/Linux podemos ter vários Clientes Gráficos. Depois que o servidor gráfico já está instalado e configurado, só vamos ter o trabalho de instalar os clientes gráficos.

Instalar o gerenciador de janelas "WindowMaker":

```
# aptitude install wmaker
```

2) Agora vamos iniciar o nosso cliente gráfico que acabamos de instalar:

```
# startx
```

3) Para o próximo teste, vamos instalar outro cliente gráfico que é muito utilizado, o "KDE":

```
# aptitude install kdebase
```

4) Depois vamos iniciar nosso outro gerenciador de janelas:

```
# startx
```

Note que foi utilizado o mesmo comando para iniciar tanto “WindowMaker” quanto o “KDE”: o “startx”. Isso acontece porque ao instalarmos o “KDE” ele se definiu como sendo o “WM” padrão do sistema, mas isso pode ser alterado.

Podemos editar o arquivo “/root/.xinitrc” para escolhermos qual cliente gráfico será iniciado quando o usuário “root” utilizar o comando “startx”. Essa configuração é válida apenas para o usuário “root”, pois alteramos o “xinitrc” da “home” do “root”:

```
1  # vi /root/.xinitrc
```

- WindowMaker utilize: **“wmaker”;**
- KDE utilize: **“startkde”.**

Para que alteração seja válida para qualquer usuário, devemos editar o arquivo de configuração global:

```
1  # vi /etc/X11/xinit/xinitrc
```

Lembrando que uma configuração local, ou seja, o arquivo pessoal do usuário, prevalece sobre o global, caso o usuário especifique um. Vamos instalar o pacote do “XFCE”:

```
1  # aptitude install xfce4
```

17.7.3 Display Managers

Vimos no tópico anterior como iniciar o nosso cliente gráfico utilizando o comando “startx”, mas isso nem sempre é muito prático. Para facilitar esse processo, podemos utilizar os chamados “Display Managers”.

Vamos instalar o “xdm”, que é bem simples:

```
1  # /etc/init.d/xdm start
```

Vamos instalar o “kdm”, que possui mais recursos:

```
1  # aptitude install kdm
```

Serão feitas algumas perguntas sobre qual será o seu “Display Manager Default”, o “kdm” ou “xdm”? Escolha sempre o “kdm”, pois dessa maneira toda vez que o seu sistema iniciar, ele vai ativar automaticamente o “kdm” no terminal 7, por padrão.

Para iniciar o “kdm” é igual. Lembrando que os demais display managers devem estar parados!

```
1  # /etc/init.d/kdm start
```

Se quiser mudar o seu “Display Manager” padrão, basta editar o seguinte arquivo:

```
1  # vi /etc/X11/default-display-manager
2 /usr/bin/kdm
```

Por fim, vamos conhecer outro “DM”, o “GDM”, padrão do “GNOME”:

```
1 # aptitude install gdm
```

 No RedHat o “Window Manager” Padrão é o “GNOME”.

 No Xorg o arquivo de configuração é o /etc/X11/xorg.conf.

17.7.4 Servidor X Remoto

O **“Xterminal”** é um recurso dos servidores gráficos X presentes em todos os computadores com GNU/Linux. Este recurso possibilita que uma máquina com menor desempenho possa executar uma aplicação gráfica a partir de um servidor, onde toda a carga de processamento é “depositada” nele, e a nossa estação atua somente como um terminal.

 O “Xterminal” utiliza o protocolo “XDMCP”.

Utilizaremos o Display Manager “gdm” para fazer esse serviço. Vamos editar o arquivo onde ativaremos o “XDMCP” para o “gdm3”:

```
1 # vim /etc/gdm3/daemon.conf
```

(servidor)Localize o bloco [security] e [xdmcp], utilizado para configuração desse protocolo. Ao encontrar esse bloco, ative o “XDMCP” inserindo “Enable=True”:

```
[security]
DisallowTCP=false

  [xdmcp]
Enable=true
```

(servidor gráfico)Reinicie o gdm3:

```
# /etc/init.d/gdm3 stop
# /etc/init.d/gdm3 start
```

(servidor gráfico)Habilite quem pode acessar o seu servidor X:

```
# xhost +192.168.0.1
```

(servidor gráfico)Caso queira desabilitar o acesso:

```
# xhost -192.168.0.1
```

(cliente)Em outra máquina exporte seu DISPLAY, ou seja, sua saída gráfica:

```
# export DISPLAY=192.168.0.1:0.0
```

Onde: 192.168.0.1:0.0 -> IP:display.screen

(cliente)Execute um programa e veja ele abrindo no servidor:

```
 # gedit
```

(cliente)Útil seria poder usar o recurso do servidor em uma máquina mais antiga:

```
1 # X -query 192.168.0.1 :1
```

Acessibilidade

17.8 Tecnologias de assistência

As tecnologias de assistência são implementadas aos computadores para torná-los mais acessíveis. Alguns DAE's (Dispositivos Automáticos de Entrada) comuns incluem:

Ampliadores de tela:são úteis para pessoas com baixa visão, funcionam como uma lente de aumento. Pode-se controlar qual área da tela querem ampliar, assim como mover o foco dinamicamente. Os ampliadores também são conhecidos como lupas ou programas que fazem uma cópia grande.

Leitores de tela: esses DAE's tornam a informação disponível com o recurso de leitura, eles "leêm"os dados contidos na tela.

Podem também traduzir gráficos, se houver um texto alternativo que descreve as imagens visuais. Leitores podem também ser úteis para pessoas com dislexia.

Ferramentas de teclado:usados por aqueles que possuem algum problema para datilografar e controlar o mouse. Por meio deste, é possível executar seqüências complicadas em série (por exemplo: ctrl + alt + del), controlar o ponteiro do mouse e as teclas do teclado.

Sintetizadores de voz: tais recursos permitem emitir voz, geralmente baseando-se em algum texto.

Dispositivos de entrada alternativos: por meio destes, é possível controlar o computador com outros meios que não sejam um teclado padrão ou um mouse. Os exemplos incluem teclados menores ou maiores, dispositivos controlados pelos olhos e pela respiração.

17.9 GOK (GNOME ONSCREEN KEYBOARD) - teclado virtual do Gnome

O teclado virtual é um software que permite entrada de texto em programas de computador de maneira alternativa ao teclado convencional. A maioria se baseia em receber cliques do dispositivo apontador (mouse) sobre uma imagem de teclado. A imagem clicada é convertida para um caracter de texto, que é escrito na tela do editor

O software GOK é livre e segue as normas de distribuição e uso da GNU LGPL (Lesser General Public License).

O projeto GOK visa permitir aos usuários acessar todas as funções do computador sem que, para isso, necessite de interagir com o mouse e/ou teclado. Ou seja, como vários usuários possuem limitação nos movimentos voluntários, o acesso às funções do computador é realizado usando métodos alternativos de entrada. Tais métodos de entrada podem ser controlados pela ação de movimentar os olhos, a cabeça, os lábios ou contrair os músculos.

Desta maneira, todos os usuários podem ter acesso universal às aplicações e funcionalidades do GNOME.

Faça o login como root

```
1 # apt-get install gok
```

Depois de efetivada a instalação, é possível executá-lo por meio de:

```
1  Aplicações -> Acessibilidade -> Teclado Virtual
```

ou

```
1  # gok
```

Caso o suporte para tecnologias assistivas não estiver liberado no seu sistema quando começar, esta tela aparecerá:

Clique em "Enable and Log Out"para habilitar o suporte de tecnologias assistivas e então fazer uso de todas as funcionalidades do seu GOK. Neste caso, é necessário iniciar o GOK novamente.

Na tela principal:

Por meio das teclas dessa janela, podem ser ativadas várias opções:

Redigir: disponibiliza um teclado virtual;

Janela: disponibiliza teclas para configurar as janelas abertas;

Mouse: disponibiliza teclas para emular controle do mouse;

Lançador: disponibiliza acesso a outros aplicativos;

Ativar: permite que se altere entre as aplicações correntes;

17.10 ORCA

Orca é um leitor de tela. Tudo o que está na tela ele lê e diz através do alto-falante do computador.

```
1 # apt-get install gnome-orca
```

Sistema -> Preferências -> Tecnologias assistivas

```
1 # apt-get install emacspeack
```

O mesmo que o Orca, ou seja, provê uma interface em modo texto para usuários que tem problemas visuais que é usado como leitor de tela.)

```
1 # apt-get install xkbset (keyboard accessibility - AccessX - X11
```

Modifica opções a acessibilidade do teclado como repetir caracteres ao apertar um tecla, a velocidade disso, apertando teclas quase que simultâneos o que deve ser feito etc. Ver: "/usr/share/X11/xkb/compat/accessx")